(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

EQUIPMENT READINESS CODES EXPERT SYSTEM
USING JOSHUA
FOR U.S. ARMY COMBAT DEVELOPMENT

by

Thomas Edward Chamberlin

June, 1990

Thesis Advisor: Se-Hung Kwak

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): 52	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Naval Postgraduate School; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code): Monterey CA 93943	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
EQUIPMENT READINESS CODES EXPERT SYSTEM USING JOSHUA FOR U.S. ARMY COMBAT DEVELOPMENT

12. PERSONAL AUTHOR(S)
Chamberlin, Thomas Edward

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM TO	June 1990	97

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Equipment Readiness Codes, Combat Development, Table of Organization and Equipment, TOE

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Expert systems have arrived as a popular productivity tool in business, industrial and managerial environments. Such tools should be extensively employed into the U.S. Army environments as well. In this thesis, an example of an expert system and its interface is presented. The expert system created, EQUIPMENT READINESS CODES EXPERT SYSTEM, enables a U.S. Army Combat Development analyst to utilize expert system technology. The advantages achieved are maintaining consistent and accurate Army Combat Development policy, reduction of the tedious, analytical tasks to the power of the machine, and the centralization of expert system maintenance and rule production. Furthermore, this expert system provides the much needed but often *scarce* expertise to ensure qualitative performance from nonexperts, provides efficiency and consistency of the experts, and even furnishes a training vehicle for others who need to understand the expert's thought process.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Se-hung Kwak	22b. TELEPHONE (Include Area Code): (408) 646-2168; 22c. OFFICE SYMBOL: 52KW

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

Approved for public release; distribution is unlimited.

Equipment Readiness Codes Expert System

Using Joshua

for U.S. Army Combat Development

by

Thomas Edward Chamberlin
Captain, United States Army
B.S., Virginia Polytechnic Institute and State University, 1980

Submitted in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
JUNE 1990

Author: _____
Thomas E. Chamberlin

Approved by: _____
Se-Hung Kwak, Thesis Advisor

_____
Major George Thurmond, Second Reader

_____
Robert B. McGhee, Chairman,
Department of Computer Science

ABSTRACT

Expert systems have arrived as a popular productivity tool in business, industrial, and managerial environments. Such tools should be extensively employed into the U.S. Army environments as well. In this thesis, an example of expert system and user interface development is presented. The expert system created enables a U.S. Army Combat Development analyst to utilize expert system technology. The advantages achieved are maintaining consistent and accurate Army Combat Development policy, reduction of tedious, analytical tasks to the power of a machine, and the centralization of expert system maintanence and rule production. Furthermore, this expert system provides the much needed but often *scarce* expertise to ensure qualitative performance from nonexperts, provide efficiency and consistency of the experts, and even furnish training for others who need to understand the expert's thought process.

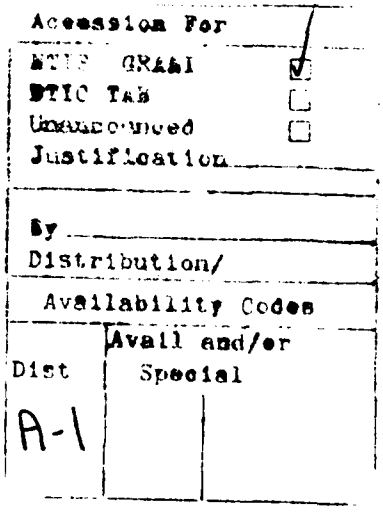

TABLE OF CONTENTS

I. INTRODUCTION

This project presents a solution to the objective set by Deputy Chief of Staff for Operations and Plans(DCSOPS), Headquarters, Department of the Army(HQDA), to develop an Equipment Readiness Codes(ERC) expert system. This system is developed on a 3650 Symbolics LISP machine running Common LISP. The expert system is effected using *Joshua*[Ref. 1:p. 1], a Symbolics, Inc. software product specifically designed to construct and deliver expert system applications. In addition, an interface is incorporated utilizing the Symbolics Common Lisp *Define-Program-Framework* Flavor[Ref. 2:pp. 21-46] to create an interactive environment between the user and the knowledge represented in the Joshua database. Through use of this expert system, a Table of Organization and Equipment(TOE) analyst is freed from the task of determining and/or reviewing ERC codes while constructing a TOE. Therefore, maximal effort is then placed on the primary task of analyzing organizational functionality, personnel requirements, and equipment requirements and usage within the specified organization.

A. WHY JOSHUA?

The first issue at hand in implementing the expert system is to provide an environment which maintains both the knowledge and rule network. Joshua is a very compact system, organized around 30 core functions and contains built-in facilities for application development. The Joshua system is coherent and straightforward due to the following three traits:

1. The syntax is LISP-like, uniform and statement-oriented, so that LISP programmers are not require to learn a new language.

2. The interface to any database is simple, consisting of only three functions, **ask**, **tell**, and **clear**.

3. Joshua contains special Zmacs facilities, such as bracket matching to ease program and rule development.

Modularity and accessibility are notable strengths of Joshua, allowing for user interfaces, control structures, and storage structures - all of which can be customized to the particular application. External databases are accessible; existing software tools can be integrated into the Joshua application; performance can be fine-tuned. The Joshua system is extensively addressed in Chapter 4.

B. SYSTEM INTERFACE

The interface is the second issue addressed in this expert system development. This interface is not intended for the general user, but rather for one qualified to maintain the rules within the expert system. The Flavor Design-Program-Framework is a extremely useful tool to develop such an interface. In addition, a Symbolics LISP machine provides an interactive code-building facility, *Frame-Up Layout Designer*, to aid in writing an interface for an application program. More specifically, Frame-Up Layout Designer is an interactive version of Design-Program-Framework[Ref. 2:pp. 103-134]. The interface provides the Joshua commands to interact with and monitor the Joshua database, as well as commands to interactively select and manage a TOE through the expert system.

C. SUMMARY OF THE CHAPTERS

Chapter 2 presents the background information and the motivation behind this effort to implement the ERC EXPERT SYSTEM. Chapter 3 presents the detailed problem statement. The ERC EXPERT SYSTEM is described in detail within Chapter 4. Chapter 5 focuses on the interface, the Define-Program-Framework Flavor, and examines the specific, previously undocumented procedures required to generate a functional interface. The summary and the conclusions found during development of this project are contained in Chapter 6.

II. BACKGROUND INFORMATION

A. OBJECTIVE

The objective of this study is two-fold: first, to develop an expert system to determine Equipment Readiness Codes(ERC) in a Table of Organization and Equipment(TOE); and second, to insure that the expert system developed integrates easily into the environment being used at the Organizations Directorate(ORGD), Deputy Chief of Staff for Combat Development(DCSCD), Headquarters Training and Doctrine Command(HQ TRADOC), Fort Monroe, Virginia.

B. MOTIVATION

There are two underlying motivations for this study. First, DCSOPS, HQDA established an objective to develop an ERC rule-based system. Such a system was developed and the results of that development are published in the U.S. Army Concepts and Analysis Agency Study Report, CAA-SR-88-14[Ref. 3:pp. 1-47]. This system was designed for DCSCD, HQ TRADOC. However, the host machine noted in the CAA Study Report was found impractical due to memory limitations and problems found in the shell environment[Ref. 1:pp. 1-3]. This study solves these problems.

Secondly, even as the CAA ERC Rule System was under development, personnel within DCSCD, HQ TRADOC were proposing the development of expert systems possessing a different system environment to further automate and revolutionize the

documentation process currently employed. An ERC expert system with a comparable environment was required. Again, the expert system developed under this study satisfies this need and integrates into the current system environment at HQ TRADOC.

C. TABLES OF ORGANIZATION AND EQUIPMENT (TOE)

A Table of Organization and Equipment is the Army's requirements document specifying a unit's mission, organizational structure, and the minimum mission essential personnel and equipment requirements necessary for that unit to accomplish its overall wartime mission. The governing document that entails the development of a TOE is Army Regulation(AR) 71-31. A TOE document goes through a series of developmental steps or levels before final approval for the document is given by HQDA.

The initial step occurs at the TRADOC proponent or school level. The designated proponent or school depends on the proposed "type" of unit to be developed. For instance, an armor unit would be generated by the U.S. Army Armor School and Center, Fort Knox, Kentucky; an artillery unit would be generated by the U.S. Army Field Artillery School and Center, Fort Sill, Oklahoma; and so forth.

Further TOE coordination and development occurs at the next level, the integrating center. An example of an integrating center is the Combined Arms Center at Fort Leavenworth, Kansas. Here, all lower level development points, in this case, combat schools, the Armor School, Infantry School, Artillery School, etc., send TOE documents for coordination and approval. After this level, the TOE document goes under a final

review and subsequent approval at ORGD, DCSCD, HQ TRADOC prior to final approval at HQDA.

At each level during the TOE developmental process, personnel and equipment are placed into the organizational structure of the unit. Also at each level, combat readiness requirements are determined. The following section reviews the methodology used to determine readiness codes.

D. EQUIPMENT READINESS CODES (ERC)

As noted above, the ERC are assigned by the TRADOC service schools as part of the TOE documentation process. Assignment is based on judgement of the TOE developer documenting the individual TOE. This judgement is guided by the governing document for Equipment Readiness Codes, AR 220-1[Ref. 4:p. 1]. By definition, codes are assigned in the following fashion[Ref. 4:App. B]:

1. **ERC-A or ERC-P (Primary Weapons and Equipment)** - The equipment directly essential to accomplishment of assigned unit missions and/or directly providing means to generate unit capabilities in a TOE. ERC-A items can be "upgraded" to a designation of ERC-P, ***pacing items***. Pacing items are equipment items which either have high-dollar values or are considered major weapon systems. Examples are a tank or an attack helicopter.

2. **ERC-B (Auxiliary Equipment)** - Equipment which supports the primary equipment noted above. This equipment may also replace primary equipment should such

equipment to become inoperative. Examples are a back-up radio or a device used to mount a weapon.

3. **ERC-C (Administrative Support Equipment)** - Equipment supportive to assigned operational missions and tasks performance. Examples are a training device or a wrist watch.

The above guidelines provide the TOE developer with the means with which to determine a readiness code for each piece of equipment in the TOE.

E. SUMMARY

The Combat Analysis Agency generated a rule-based system to meet the objective of DCSOPS. However, their system has fundamental problems with the memory capacity of the host machine and the shell environment under which it was developed. In addition, system integration into the expert system environment in progress at HQ TRADOC may have been difficult. The following chapter details the specific problems addressed in generating an adequate ERC expert system to meet the DCSOPS' objective.

III. DETAILED PROBLEM STATEMENT

A. INTRODUCTION

An expert system for this thesis, **ERC EXPERT SYSTEM**, is created to revise the CAA's pc-based ERC Rule System and allows integration into ongoing HQ TRADOC TOE expert system development. The ERC EXPERT SYSTEM rules are written using Symbolics, Inc. software, **Joshua**; and the system's interface is developed using Symbolics Common LISP, **Define-Program-Framework** Flavor[Ref. 2:pp. 25-27]. All system functions are written in LISP, and can be executed on any Symbolics 36xx family of computers. Discussed below are the three key issues investigated in order to resolve the design and implementation issues of this expert system: the knowledge-representation, the inference mechanism, and the system interface.

B. KNOWLEDGE REPRESENTATION

Historically, expert systems capture the knowledge or expertise of an "expert", so that others may benefit from that person, particularly in the absence of that person. The challenge is to capture that knowledge in a manner useable to the system in which it is designed. Thus, knowledge representation is a key component in developing an expert system. Bowerman and Glover discuss and categorize three schemes to represent knowledge: *rule-based*, *frame-based*, and *object-oriented* representations[Ref. 5:p. 100]. Nielsen and Walters also address the first two schemes noted above, but present other

techniques: *multiple contexts*, *model-based*, and *blackboard* representations[Ref. 6:pp. 195-319]. By far the most widely used method is rule-based, originally established by the production rule language **OPS5**. The rule-based approach places emphasis on very shallow knowledge and places all parameters of the knowledge into a single production rule. Rules within this scheme take the form of IF/THEN or IF/THEN/ELSE statements. Appendix I of the CAA report[Ref. 3:pp. 1-134] shows the rules of that system represented in this fashion. The other strategies of knowledge representation are applicable when representation of more indepth knowledge is needed. In this case of the readiness codes, a very simple one-to-one relationship is seen between the equipment situation and ERC assignment. Since the scope of this study can neither readdress the entire design phase, nor provide access to the experts for interviewing, the knowledge representation scheme used in Module I is acceptable and appropriate to represent the knowledge within this study's expert system.

C. INFERENCE MECHANISM

The second issue analyzed is the inference method. Here, the TOE analyst provides the answer. In an analyst's daily work of developing a TOE, a piece of equipment is placed into a unit to satisfy a mission requirement and/or accomplish a specified task. In turn, equipment readiness codes are assigned in relation to the degree to which that piece of equipment is critical to the unit's overall mission and tasks. Therefore, a unit's mission and specified tasks determine the readiness state for that unit, as well as the personnel and equipment requirements/readiness within the unit's structure. In other

words, the system can inference forward from the known predicates or facts (unit mission and tasks list) in order to derive as many consequences as possible (readiness status/codes for personnel and equipment). This leads to an excellent example of a data-driven or forward chaining approach to inference on the knowledge base, also known in logic as *modus ponens reasoning*[Ref. 7:p. 102].

D. SYSTEM INTERFACE

The final concern addressed is the system interface. The chosen expert system development environment at HQ TRADOC is Common LISP. Therefore, this study also requires a Common LISP environment. Due to Joshua's strengths, it is selected as the application software for the rules and management of the knowledge. In addition, Joshua allows Common LISP code anywhere within its system structures. In fact, Common LISP functions are allowed and even expected within the structure of Joshua rules. Lastly, Symbolics' Define-Program-Framework Flavor is the tool of choice to provide the necessary visual window interface specifically for display of the database, rule firing, and system commands. This also coincides with the windowing environment of HQ TRADOC in their system developments.

E. SUMMARY

This chapter discussed the issues faced during the course of this study. For program implementation, Common Lisp was adopted as the system environment. In addition, rule-based knowledge representation, forward chaining inference, the Symbolics software package Joshua, and the windowing features in Define-Program-Framework round out the

complete environment used within this thesis. Using these conditions, the ERC EXPERT SYSTEM is examined in the following chapter.

IV. ERC EXPERT SYSTEM

A. INTRODUCTION

The **ERC EXPERT SYSTEM** is designed to generate ERC codes for all equipment items within any given TOE, i.e. the TOE needs not be a DA approved TOE. The rules to generate these ERC codes are derived from CAA Study Report, Appendix I, CAA-SR-88-14[Ref. 3:pp. 1-134]. Thus, this system can be used during TOE development prior to the HQ TRADOC review board process and eventual DA approval. ERC EXPERT SYSTEM consists of two parts: the expert system itself and the interface. The expert system and its development in Joshua are discussed within this chapter and the interface is discussed in Chapter 5.

B. JOSHUA

1. OVERVIEW

Joshua is a software product implemented on the Symbolics 36xx family of computers operating under the Genera operating system environment[Ref. 1:p. 1]. Joshua is integrated with LISP, and allows LISP code to be used within user-defined rules. The Joshua software itself is implemented with Flavors[Ref. 1:p. 1]. This allows flexibility in the ERC EXPERT SYSTEM. The flexibility is greatly seen in the use of Common LISP code throughout the application developed, and within Joshua structures, such as rules. This allows for message displays not related to Joshua, but rather to the application itself.

In addition, external structures, such as databases, can be addressed and manipulated directly using LISP functions and methods.

The heart of Joshua is a rule-based inference language, consisting of five major elements:

1. ***Predications*** - The knowledge or facts of the database; also referred as statements, or assertions.

2. ***Rules*** - The means of defining relationships among predications, as well as procedural knowledge.

3. ***Database*** - The entire collection of predications and rules remembered by the system.

4. ***Protocol of Inference*** - Joshua's mechanism to integrate the above elements and execute the system reasoning.

5. ***Truth Maintenance System (TMS)*** - The system to maintain explanations of the reasoning determined by the Protocol of Inference, and maintain overall system consistency.

Modularity is designed within Joshua. This modularity localizes developmental changes, and supports system modeling[Ref. 1:pp. 1-6], and thus, the Protocol of Inference is further divided into five functional groups:

1. ***Database Interface*** - Supervises additions/deletions of predications to/from the database.

2. ***Truth Maintenance System (TMS) Protocol*** - Supervises all deductive dependencies.

3. ***User Interface Protocol*** - Controls the interaction with a system user.

4. ***Rule Indexing Protocol*** - Manages the rules in the database.

5. ***Rule Customization Protocol*** - Manages the rule compiler.

Additionally, Genera's program development facilities are also available for use by Joshua, to include the Zmacs editor, User Interface Management System, input/output abilities, and Dynamic Windowing.

2. JOSHUA PREDICATIONS

a. Introduction

Joshua, like other AI programming facilities, entails working with facts that represent the knowledge of the system. Programs then build, store, and reason with these facts, going on to build, store and even remove other new facts. This cycle then begins again until no new facts are available. Joshua facts, called *predications*, are really just Flavor instances. This section discusses the essentials of creating and storing predications. Other related topics, such as logic variables and logical connectives, are also introduced. The process of using predications for reasoning is within the realm of rules and is reviewed in the section 3.

b. Predications and Predicates

Predications are stored in the expert system database. The knowledge they represent is available to the system and can be manipulated by system rules. Broadly equivalent terms for predications are *statements*, *assertions*, and *facts*. The Joshua system protocol also allows additional system manipulations other than just rule manipulation[Ref. 1:p. 13]:

1. Insertion of predications into the database.
2. Review of predications in the database.

3. Conduct inference of predications by system rules.

4. Supply of specific justifications to predications.

5. Deletion of predications from the database.

Note, predications can be manipulated like any other LISP object. Predications consists of two parts - the *predicate*, the first item, and its corresponding arguments, zero or more. Brackets always enclose a predication. Figure 4.1 shows some examples of predications.

```
[healthy Catherine]
[author-of (poems plays) Shakespeare]
[is-assigned current-lin erc-a]
```

Figure 4.1. Predication Examples

c. *Defining Predicates*

Predicates are the names of relationships; they organize knowledge within a predication and further express the relationship among its parts. A Joshua predicate must be defined prior to its use within a predication. The system macro **define-predicate** is the vehicle to define new predicates. This definition establishes the format, specifies the required arguments within the predication in which it is used, and optionally shows any customized method of controlling the predicate. To remove any predicate definition, the system macro **undefine-predicate** must be used. It is important to note that while this macro removes the predicate definition, it does not remove any predications built with that predicate prior to undefining it. These predications must be explicitly removed from the database world. Examples of predicate definitions are shown in Figure 4.2.

```
(define-predicate healthy (any-object))
(define-predicate has-night-ops (current-toe))
```

Figure 4.2. Examples of Defining Predicates

Once predicates are defined, knowledge can be expressed in numerous different predications, each using different arguments depending on the context of the specific problem. The ERC EXPERT SYSTEM predicates are found in Appendix L. These predicates are the first items loaded upon system load into the database, and are immediately available for use in predications. Using the above definition of the predicate "healthy", predications can be made as shown in figure 4.3.

```
[healthy eat-fruit]
[healthy exercise-regularly]
[healthy Mary]
```

Figure 4.3. Predications Using "healthy" Predicate

Using the context of the predicate **healthy** in above figure, if the problem required the identification of healthy people then the example concerning Mary would be appropriate. In another problem, identification of ways to maintain health, the predications of eat-fruit and exercise-regularly are suitable. Overall, the same predicate definition can be used in numerous applications.

d. Predications and Truth Values

For any predication to be "available" to the system, it must be inserted into the database. The function **tell** asserts a predication into the database. The database is the extensible collection of all predications and their associated information. The function **untell** removes a predication from the database, and frees any related storage space as well. As the name implies, **untell** is the opposite function of **tell**. Using **tell** is quite simple, as shown in Figure 4.4.

```
(tell [has-night-ops 17487L000])
[has-night-ops 17487L000]
T
```

Figure 4.4. Use of the **tell** function

Note that **tell** returns the predication object that is asserted into the database and a boolean value whether the predication is being inserted for the first time or not. This is not the associated truth value. If the above predication were previously inserted into the database, the boolean value **nil** would be returned. A truth value denotes what the system knows about the truth state of any database predication; it is associated with a predication when the predication is inserted into the database. The truth values of predications change as knowledge is acquired and the database is updated. Predications can have only one of four possible truth values:

1. ***true*** (appears under "True things" in database display).
2. ***false*** (appears under "False things" in database display).
3. ***unknown*** (does not appear in database display).

4. ***contradictory*** (a transient state; does not appear).

Truth values are manipulated by the user or by the **Truth Maintenance System(TMS)**. The TMS is covered in a future section.

Joshua has a three-valued logic system. A predication is ***true*** if its arguments are believed to satisfy the predicates, and ***false*** if its arguments are not. When using predications, the function **tell** applies the belief that a predicate's arguments are true. The **not** prefix to a predication changes the truth value of a predication from ***true*** to ***false***, and vice versa. It is very important to understand that the **untell** function does not reverse a truth value, but rather deletes a predication from the database. Examples of the use of **tell**, **untell** and **not** follow in Figure 4.5.

If a predication is neither ***true*** nor ***false***, it is ***unknown***. A predication becomes ***unknown*** when no valid reason supports it. In some languages, such as Prolog, a fact is assumed to be false until proven to be true. Joshua does not subscribe to such a "closed world view". For example, if a predication's truth value is originally ***true***, and the underlying reasoning for its truth is removed from the database, the predication's truth value becomes ***unknown***. From a reasoning viewpoint, a predication with a truth value of ***unknown*** is indistinguishable from one that is not in the database at all. Thus, a predication whose truth value changes to ***unknown***, physically remains in the database but is conceptually not visible. If the underlying reasoning is reinserted into the database, then the predication will again be "visible"; its truth value will change back to ***true***, and the predication will once again be used in the inference process. This is efficient in the sense that predications do not require

reassertion as the database is constantly changed by system inference and reasoning. The truth value of ***unknown*** is primarily useful to the TMS to maintain logical consistency as the database is modified.

```
   (tell [has-night-ops 17487L000])
   [HAS-NIGHT-OPS 17487L000]

   (tell [has-night-ops 06203L000])
   [HAS-NIGHT-OPS 06203L000]

   Show Joshua Database
True things
   [HAS-NIGHT-OPS 17487L000]
   [HAS-NIGHT-OPS 06203L000]
False things
   None



   (tell [not [has-night-ops 17487L000])
   [NOT [HAS-NIGHT-OPS 17487L000]

   Show Joshua Database
True things
   [HAS-NIGHT-OPS 06203L000]
False things
   [HAS-NIGHT-OPS 17487L000]



   (untell [not [has-night-ops 17487L000])
   NIL

   Show Joshua Database
True things
   [HAS-NIGHT-OPS 06203L000]
False things
   None
```

Figure 4.5. Use of **tell**, **untell**, and **not**

The final truth value for a predication is ***contradictory***. This occurs when a predication is believed to be both ***true*** and ***false*** at the same time. This truth value is also primarily meaningful to the TMS. An excellent example of a contradiction is reviewed below and is found in the Joshua Basics Manual[Ref. 1:p. 20]. Using the mythical story about Medea and her son Jason, two **tell** operations insert facts about Medea. The first fact is a direct input into the database, **(tell [loves Medea Jason])**; the second fact, **[LOVES MEDEA HER-CHILDREN]** is deduced from some forward chaining rule, based on the belief that **[loves Medea Jason].** Now, if the first fact's truth value is changed by using the **not** prefix, the fact becomes ***false***, but the second fact that was generated by the rule remains, and thus a contradiction is generated.

e. Logic Variables

A Joshua logic variable is a special object and is recognized by Symbolics Common Lisp. A logic variable is identified by the equivalence symbol ≡. In contrast to constants, logic variables provide the ability to make more generalized statements and queries about predications, and provide capability to generate patterns. Figure 4.6 demonstrates the use of logic variables and pattern matching. Presented first are some statements about children using the "child" predicate. Then using the Joshua system query function **ask**, with a logic variable, all of the children in the database can be found. The query with this logic variable finds all correct matches within the database. Figure 4.6 also shows the query concerning the assertions of the children into the database. The query is invoked three times, once for each pattern that satisfies the query. An explanation of this query follows. At the query onset, the logic variable **≡person** is

```
(define-predicate child (person))

(tell [child Catherine])
[CHILD CATHERINE]
T

(tell [child Joe])
[CHILD JOE]
T

(tell [child Chris])
[CHILD CHRIS]
T



(ask [child ≡person] #'print-query)

[CHILD CATHERINE]
[CHILD JOE]
[CHILD CHRIS]
```

Figure 4.6. Logic Variables and Pattern Matching

uninstantiated and can match any database object. Therefore, this logic variable matches any argument in the same position in the database predication, as long as the predicate used matches the predicate **child** used in the query. At this point, Joshua searches the database to find the first predication with the predicate **child**. The first predication found is the one with the argument **Catherine**. Therefore, **Catherine** is temporarily instantiated for the logic variable **≡person**. The query pattern matches and the resulting answer is printed. Once the query is executed, Joshua uninstantiates the logic variable and searches again until all matching patterns are exhausted.

f. Logical Connectives

Up to this point a predication has been discussed simply as a single predicate and its corresponding arguments. In general, knowledge statements are much more useful when expressed in some logical combinations with other statements. Joshua provides three logical predicates **and, or**, and **not** to accomplish this task. All of these connectives are extremely important to the develop of the ERC rules. Inability to use these connectives in the antecedent portion of the rule is another flaw in the expert system shell in the CAA report. Once predicates are defined, compound predications can be created as shown in Figure 4.7.

```
(tell [and [child Mallory]
           [child John]])
[CHILD MALLORY]
[CHILD JOHN]



(tell [and [has-night-ops 17487L000]
           [has-night-ops 06108L000]]
           [has-cat-code 17487L000 1])
[HAS-NIGHT-OPS 17487L000]
[HAS-NIGHT-OPS 06108L000]
[HAS-CAT-CODE 17487L000 1]



(ask [and [or [child ≡person]]
          [and [has-night-ops ≡toe-num]
               [has-cat-code ≡toe-num 1]]]
    #'print-query)

[CHILD MALLORY]
[CHILD JOHN]
[HAS-NIGHT-OPS 17487L000]
[HAS-CAT-CODE 17487L000]
```

Figure 4.7. Logical Connectives/Compound Predications

Overall, logical connectives ease the assertion of data into the database, and additionally focus and/or refine queries. This in turn cuts down on database search time.

g. Summary

Within this section the basic establishment of the knowledge database was discussed. The foundation to the ERC EXPERT SYSTEM database is the predication or fact. Predications have four possible truth values, ***true***, ***false***, ***unknown***, and ***contradictory***. The use of logical variables and logical connectives enhances the manipulation of predications created in the database. The following section about Joshua rules details how an expert system can reason on the predications in the database.

3. RULES AND INFERENCE

a. Introduction

A rule is an independent composition of declarative and procedural information that defines how a system conducts inference[Ref. 1:p. 37]. Inference is the process by which an expert system drives through a set of given rules, acquiring new facts during the process, and executing rules to arrive at a conclusion[Ref. 3:p.3-1]. So far, predications have been identified as the way to define and collect needed information for the expert system. Predications within the ERC EXPERT SYSTEM provide the database with facts about specific pieces of information in a given TOE. ERC rules define the *reasoning* or *inference process* that can be made from these known equipment facts. As noted earlier, the rules developed for the ERC EXPERT SYSTEM are derived

from Appendix I of the CAA Study Report[Ref. 3:pp. 1-134]. In this section, Joshua's rule control structure, rule definition, how rules work and rule monitoring are discussed. From this awareness of the Joshua rule system, ERC rules are easily created.

b. Forward Verses Backward Chaining

Reasoning by rules in Joshua entails either forward or backward chaining[Ref. 1:p. 37]. Forward chaining is *data-directed inference*, reasoning from known facts to some conclusion[Ref. 1:p. 37;Ref. 7:p. 102]. In Joshua, forward chaining is activated by **tell.** Thus, whenever a new predication is asserted into the database, the system examines the forward chaining rules, and reasons to derive conclusions from the new knowledge given by the tell statement. Backward chaining, on the other hand, reasons to satisfy some given conclusion. Backward chaining is defined as *goal-directed inference*[Ref. 1:p. 37;Ref. 7:p. 100]. A backward chaining rule looks for facts to support a goal. In this study, the TOE provides the facts about pieces of equipment and their usage within a TOE. The rules within this expert system lead from these facts to generate the conclusion, i.e. the correct ERC. Thus, forward chaining is the control structure chosen for ERC EXPERT SYSTEM.

c. Defining Forward Rules

Joshua rules are defined with the system function **defrule**. All Joshua rules have the following parameters:

1. A user-supplied rule name.
2. A required keyword specifying the rule's control structure(either forward or backward chaining).

3. A combination of patterns divided into the *antecedent* (also known as the *trigger part*, or *if-part*), and the *consequent*(also known as the *action part*, or *then-part*).

The Joshua syntax for the function **defrule** follows[Ref. 8:p. 126]:

defrule *rule-name* (*control-structure* &rest *arguments*)
if *if-part* then *then-part*

Figure 4.8 depicts an example of a rule definition.

```
(defrule dragon-id-kit (:forward)
   if [and [huge ≡creature]
           [breathes ≡creature fire]
           [or [guards ≡creature gold]
               [guards ≡creature maiden]]]
   then [dragon ≡creature])
```

Figure 4.8. Example of a Rule Definition

In this forward rule, the *trigger part* is a compound predication pattern that must be completely satisfied for the *action part* to execute or fire. This rule shows the entire *trigger part* is joined by **and**, thus all conditions under **and** portion of the rule must be met to satisfy the forward trigger. There is also an **or** connective, which allows any condition within its portion to be met and then the **or** is satisfied. The Joshua command **Show Joshua Rules** displays all currently defined rules. This command has various options to allow for tailoring of the display[Ref. 8:pp. 133-134].

d. How Forward Rules Work

As stated previously, data-directed inference is activated by the assertion of new facts into the database with the **tell** function. A fact is only new when the system is "told" something for the first time. Once a fact is in the database, if you **tell** the same fact again, it is no longer new and will not activate any rule firing. Similarly, a fact that

is unjustified, and then you **tell** that fact again, it is not new knowledge since it was never removed from the database.

When all conditions of forward rule's *if-parts* are satisfied, the rule is then triggered; it fires and executes the rule's actions in the *then-parts*. The action part can stipulate any action to include LISP code. Any new facts inferred from the current facts are automatically asserted into the database, and in turn can trigger more rules. This can continue to generate chains of new facts and rule firings until no more new facts are generated. Using the rule definition of a "dragon-id-kit" in Figure 4.8, the following figures show an example of the rule firing. Figure 4.9 will provide the first two predications necessary to trigger the rule.

```
(tell [huge dudley])
[HUGE DUDLEY]
T

(tell [breathes dudley fire])
[BREATHES DUDLEY FIRE]
T
```

Figure 4.9. Predications for Rule Triggering

At this point, the database contains the rule definition found in Figure 4.8 and the two predications shown above. These predications satisfy the first two lines in the rule definition. However, it will require a third predication about what the "creature" guards before the rule is triggered. Figure 4.10 will add the final predication to trigger the dragon-id-kit rule. Thus, the addition of the final predication completely satisfies the rule

```
(tell [guards dudley gold])
[GUARDS DUDLEY GOLD]
T
[DRAGON DUDLEY]
```

Figure 4.10. Final Predication to Trigger Rule

trigger, causes it to fire and generates the new fact **[DRAGON DUDLEY]**, i.e. identifying Dudley as a dragon. This chain of forward chained inferences continues as long as there are new facts that fully trigger forward rules in the system.

e. Rule Tracing

To watch the execution of rules, the Joshua system command *Enable Joshua Tracing* needs to be issued. This command accepts an option of *Forward Rules*, *Backward Rules*, or *All* depending on the type of rule monitoring desired[Ref. 1:pp. 110-113]. It also presents a message every time a rule fires. With forward rules the message appears when the *if-part* of the rule is completely satisfied, and just prior to the execution of the *then-part*. Figure 4.11 displays forward rule tracing.

```
Enable Joshua Tracing (Type of tracing) Forward Rules
-----------------------------------------------------
Forward Chaining tracing is on
   Tracing All forward rules
   Traced Events: Fire and Queue
-----------------------------------------------------
  (tell [guards dudley maiden])
  [GUARDS DUDLEY MAIDEN]
 ▸ Firing forward rule DRAGON-ID-KIT (1 trigger)
  ▸ [DRAGON DUDLEY]
  NIL
```

Figure 4.11. Tracing Forward Rules

Notice that in this execution of the rule the message appeared after the predication that satisfied the rule trigger and just before the action took place. In addition, the system response of NIL appeared at the end because the fact of dudley's identification as a dragon is already known in the database.

f. Summary

This section entailed the concepts of Joshua rules and the rule inference. Forward chaining is the control structure used due to the clear dependence on the TOE data that drives the rule definition. Joshua provides a very simple format to follow to define any rule required. In addition, the Joshua system furnishes a tracing mechanism to aid in debugging programs and monitoring rule dependence. The final section of this chapter will briefly discuss the Joshua Truth Maintenance System.

4. JOSHUA TRUTH MAINTENANCE SYSTEM

A Truth Maintenance System (TMS) is a device used by deductive systems to maintain dependencies and relationships among statements or facts in a knowledge database. There are two primary functions of TMS: first, to annotate and preserve the reasoning support of all predications in the database; and second, to maintain the logical consistency and truth of the predications.[Ref. 1:p. 64; Ref. 6:p. 265] There are three major types of TMS[Ref. 6:p. 268]:

1. Justification TMS (JTMS)
2. Logic-based TMS (LTMS)
3. Assumption-based (ATMS)

Joshua provides a TMS as part of its overall software system, and is an option that can be included at the developer's discretion in any application development. The use of the TMS is included in this study. Joshua supports a logical or clausal TMS(LTMS)[Ref. 1:p. 64]. To use this LTMS, it must be provided as an argument to a predicate definition. Figure 4.12 provides a predicate definition specifying the use of the Joshua LTMS.

```
(define-predicate has-night-ops (current-toe)
                  (ltms:ltms-predicate-model))
```

Figure 4.12. Specifying the Use of LTMS

The LTMS provides the ERC EXPERT SYSTEM with the mechanism to manage the predications in the database. In addition, the LTMS operations can be traced and monitored in the same fashion as rules. The Joshua command *Enable Joshua Tracing TMS Operations* will invoke the tracing. Once activated, this tracing will display messages of all predication manipulation, truth value changes within the database and also provide the underlying reason for such a change. A complete overview of the Joshua LTMS is provided in Section 9 of the Joshua manual[Ref. 1:pp 64-78].

C. A GUIDE THROUGH THE ERC EXPERT SYSTEM

1. INTRODUCTION

The basic concepts of the expert system and Joshua have been presented. A step-by-step escort through the implementation follows. This guide details the highlights of the ERC EXPERT SYSTEM. Predicate definitions, reading of the data from a TOE

into the database, rule usage, and LISP functions used within the system are all presented. Once this review is finished, a complete understanding of the entire system should be acquired. Further development of the system can also be accomplished.

2. THE GUIDE

The predicate definitions of the ERC EXPERT SYSTEM are maintained in the LISP file ***erc-predicates***. This file and all others referenced are found in the appendices. In particular, the predicates for night operations, branch, old and new ERC, and for the current line item number(LIN) will be addressed. The definitions for each of these predicates are shown in Figure 4.13.

```
(define-predicate has-night-ops (*current-toe*)
                  (ltms:ltms-predicate-model))

(define-predicate has-branch (current-toe *branch*)
                  (ltms:ltms-predicate-model))

(define-predicate is-lin (current-lin *lin*)
                  (ltms:ltms-predicate-model))

(define-predicate has-old-erc (current-lin *old-erc*)
                  (ltms:ltms-predicate-model))

(define-predicate has-new-erc (current-lin *new-erc*)
                  (ltms:ltms-predicate-model))
```

Figure 4.13. ERC EXPERT SYSTEM Predicates

Note that in all of these predicate definitions the option to use the TMS as part of the predicate is chosen. This aids the rule developer in "seeing" what occurs in the database

in order to track rule dependencies and/or correct present rules. Once all system have been defined, assertion of data in the form of predications can occur.

The data to be asserted into the database comes from the TOE selected by the user for evaluation. After this selection occurs, either the "auto-mode" or "step-mode" command is chosen from the command menu. Both commands' functionality is the same in regards to extracting data from a TOE for instantiation of the arguments of a predicate definition. The LISP functions ***auto-read-current-toe*** and ***read-current-toe*** extract a line of data from the TOE. Each line of data is tested for the record type, i.e. A-record, B-record, etc., and the appropriate data is assigned to a specified global variable in accordance to the record type. The record layouts for each type of record are found in the appendices. The data required for the predicates being shown here are the branch for the TOE to instantiate the ***branch*** variable; the current LIN and its current ERC, if any, to instantiate the ***lin*** and ***old-erc*** variables. The ***new-erc*** and the ***current-toe*** variables will be instantiated only if the rule is triggered, causing the variable assignment to transpire. Again this action is seen within the Joshua Display of the interface because the predicates are defined with the TMS option.

There are two rules to monitor which can be affected by the predicates defined above. Both rules are shown in Figure 4.14. As each predication is asserted by the read functions noted above, the database is continually searched to match this assertion with any rule trigger. This repetitious procedure continues until no further predication assertion is made. If an entire trigger of a forward rule is matched then the rule will fire. In order to generate this rule firing here, numerous predication assertions must be made.

For example, a branch of 01, 07, 17, 19 or 31 that is part of the A-record layout for a TOE must be found in the database to fire the night operations rule. Assertions must be made into the database to execute the rule firing. The function ***show-rule-firing*** at the end of the all rules, displays the result of a rule's action. In this case the difference or concurrence between the ***old-erc*** and ***new-erc*** is found by the system. Figure 4.15 shows the effects of predications, made after reading numerous lines of a TOE.

```
(defrule night-ops131
         (:forward
          :documentation CAA report, p. I-23)
  if [or [has-branch CURRENT-TOE 01]
         [has-branch CURRENT-TOE 07]
         [has-branch CURRENT-TOE 17]
         [has-branch CURRENT-TOE 19]
         [has-branch CURRENT-TOE 31]]
  then (tell (make-predication
           '(has-night-ops ,CURRENT-TOE))))


(defrule ercp551
         (:forward
          :documentation CAA report, p. I-81)
  if [or [is-lin CURRENT-LIN T13168]
         [is-lin CURRENT-LIN T13169]
         [is-lin CURRENT-LIN T13174]
         [is-lin CURRENT-LIN Z77258]
         [is-lin CURRENT-LIN F40307]
                  *
                  *
         [is-lin CURRENT-LIN H57505]
  then [and (setf *new-erc* 'P)
            (tell (make-predication
             '(has-new-erc CURRENT-LIN *new-erc*)))
            (show-rule-firing)])
```

Figure 4.14. ERC EXPERT SYSTEM Rule Definitions

```
 (tell (make-predication '(has-branch CURRENT-TOE 17)))
▸ Justifying : [HAS-BRANCH CURRENT-TOE 17] <-- Premise
▸ Firing forward rule NIGHT-OPS131 (1 trigger)
  ▸ Justifying : [HAS-NIGHT-OPS CURRENT-TOE] <-- Rule:
                                                 Night-OPS131

 (tell (make-predication '(has-old-erc CURRENT-LIN B)
▸ Justifying : [HAS-OLD-ERC CURRENT-LIN A]
  T

 (tell (make-predication '(is-lin CURRENT-LIN K56733)))
▸ Justifying : [IS-LIN CURRENT-LIN K56733] <-- Premise
  T

 ** assertions currently have no effect on any rule **


 (tell (make-predication '(has-old-erc CURRENT-LIN A)
▸ Justifying : [HAS-OLD-ERC CURRENT-LIN A]
  T

 (tell (make-predication '(is-lin CURRENT-LIN T13169)))
▸ Justifying : [IS-LIN CURRENT-LIN T13169] <-- Premise
▸ Firing forward rule ERCP531 (1 trigger)
  ▸ Justifying : [HAS-NEW-ERC CURRENT-LIN P] <-- Premise

TOE: 17487L000 PARA: 3 LIN: T13169 OLD-ERC: A NEW-ERC: P
```

Figure 4.15. Effects of Predications on Rule Triggers

The last feature shown in this tour of the ERC EXPERT SYSTEM is the rule management. Any rule name that appears in the Joshua Display of the interface is automatically mouse-active. Therefore the rule can be pointed to and various operations can be induced on that rule. This is one of the strong points that the Joshua software provides to the expert system developer. For instance, to view a rule definition, middle-click with the mouse, the rule definition displays within the Joshua Display Window. Most important are the special key **meta** and the left-button. This key combination inserts the rule pointed at into the Z-macs editor for immediate editing. These operations are addressed in Chapter 5.

Overall, while a TOE is in the development process, a question can arise during the review of the TOE document concerning an ERC assignment. The appropriate TOE can be selected into the expert system, stepped through to identify all rule firing and rule consequences, and finally, provide on-the-spot corrections either in the rule itself or the TOE document. In either case, the ERC EXPERT SYSTEM provides an extremely powerful tool to the TOE documentation process.

D. SUMMARY

This chapter discussed the elements of the ERC EXPERT SYSTEM and the specifications to run this system on top of the Joshua software. Predications are the fundamental element to provide facts to the database. Through matching of predications to the if-part of system rules, new facts are generated or conclusions found, i.e. appropriate ERC codes are assigned to TOE equipment. The tracing capabilities of the

Joshua and its rule managing features, facilitate the supervision of the database a manageable task. Lastly, the LTMS ensures overall system consistency. This guide presented the ERC EXPERT SYSTEM's internals and demonstrated the functionality of the overall system. Using this guide as a base, further development of the ERC EXPERT SYSTEM can be accomplished. In the following chapter the interface is presented to demonstrate the overall system use and provide a "visual" point of view to the system.

V. USER INTERFACE

A. INTRODUCTION

The ERC EXPERT SYSTEM is not a stand-alone system, but is intended to integrate into the overall automated TOE documentation process. In this case, the ERC EXPERT SYSTEM is designed to read the output of a TOE created by the TRADOC DOCUMENTATION SYSTEM(TDS) database maintained at Fort Leavenworth, Kansas. The ERC coding process should follow these steps:

1. A TDS TOE document is transmitted through some communication channel to the Symbolics 36xx machine maintaining the ERC EXPERT SYSTEM.
2. The ERC EXPERT SYSTEM is invoked in the automatic mode.
3. The ERC codes are appropriately assigned by the system.
4. The updated TOE is retransmitted back to the TDS.

Once a TOE is available on Symbolics 36xx disk for retrieval into the ERC EXPERT SYSTEM, a session is ready to be performed. The following sections describe the visual interface, conduction of an ERC EXPERT SYSTEM session, and rule management.

B. VISUAL INTERFACE

The visual interface was created using the Define-Program-Framework Flavor provided within Symbolics Common LISP. Figure 5.1 shows the interface window. The expert system interface has the following features:

Figure 5.1. The Interface Window

1. A set of TOE information windows displaying all relevant TOE information for the selected TOE.

2. A Joshua display window depicting the current status of the database, assertion of facts, display messages from the Truth Maintenance System, and user selected information during rule maintenance.

3. A mouse-activated command menu.

4. A 'step' mode is provided for rule maintenance. In addition, this mode has the side effect of permitting a TOE developer to learn how and why a particular ERC code is applied to a specified piece of equipment in a TOE.

5. An 'auto' mode is provided and is the preferred method of use. User intervention is not required for this mode to run.

The interface consists of nine windows: seven display windows, a command menu window, and an interactor window. The entire windowing display is created using Symbolics Common Lisp Flavor *Define-Program-Framework*. The details to this Flavor are well documented[Ref. 2:pp. 21-46].

The upper six windows are display windows containing specific TOE information for a user selected TOE. The expert system updates these display windows automatically as the system runs. Figure 5.2 presents an example of the TOE information from left window to right:

1. The specific TOE number of the TDS document currently in the database. The TOE number is read from the A-record of a TOE document.

2. The TOE title of the current TOE document and is also read from the A-record.

3. The current TOE paragraph read into the database. The paragraph is derived from the B-record of the document.

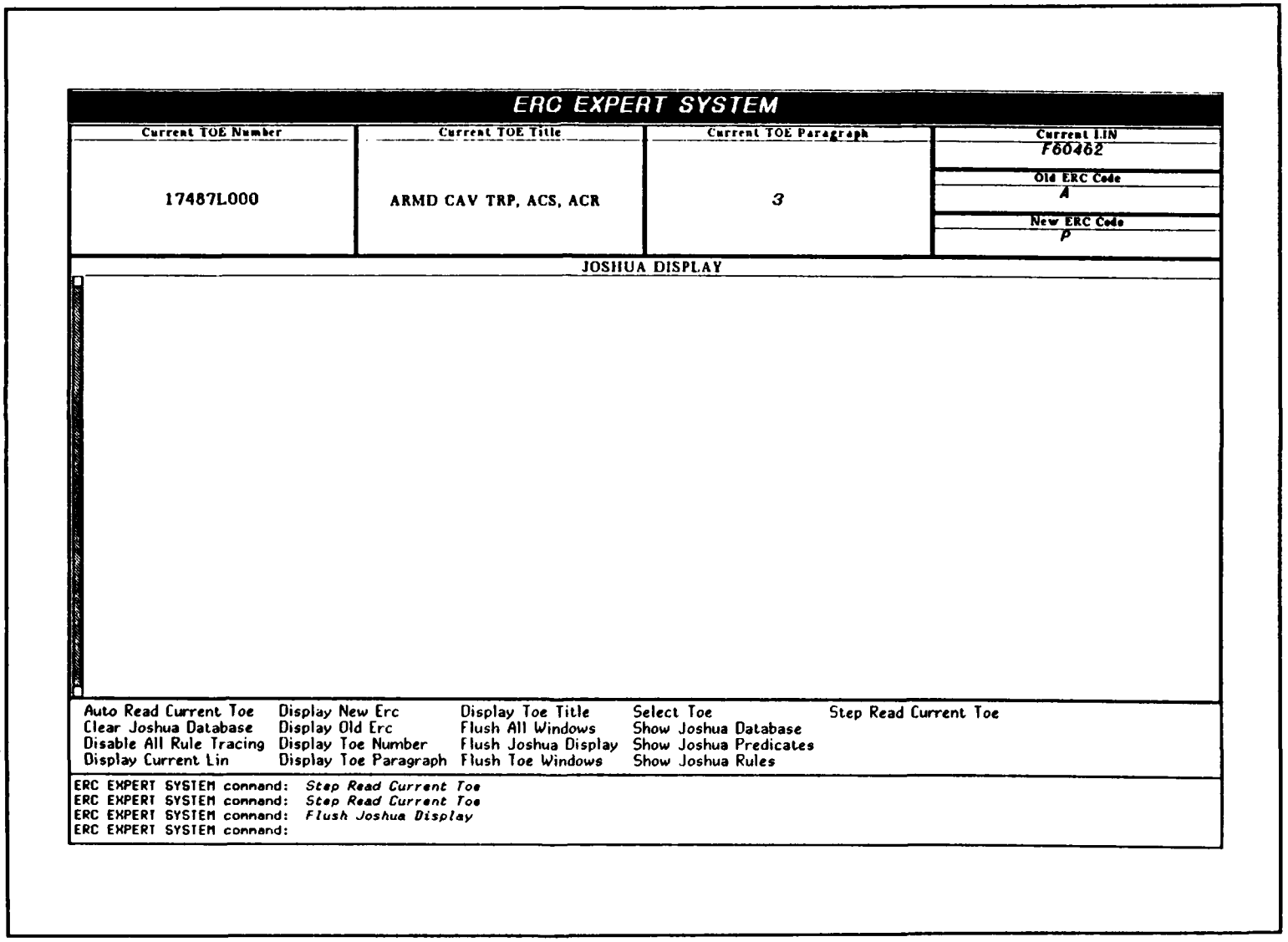

Figure 5.2. Window with TOE Information

4. The current Line Item Number(LIN) read into the database. This is the actual piece of equipment which has been read into the database and is reasoned with.

5. The "old" ERC is the current ERC code for the LIN noted above.

6. The "new" ERC is the correct ERC code determined by the ERC EXPERT SYSTEM rules for the given piece of equipment noted above.

The large main window centered in the Symbolics monitor is also a display window titled **Joshua Display**. Interactively displayed in this window are: the predications resulting from direct assertions within the code and from forward-chaining events, database information derived from execution of Joshua commands, and system generated messages. The Joshua display window is a scrolling window providing easy access to the relatively large volume of information displayed during any user session.

At the bottom of the screen are the command menu window and the interactor window. The command menu contains all expert system commands necessary to activate TOE selection, induces reading of the TOE, and retrieves Joshua database information. Note, several other system commands not displayed on the command menu are embedded into Lisp functions or consolidated into commands displayed on the menu. As the user selects a mouse-actuated command from the command menu, the selected command is displayed in the interactor window. Any resulting message and/or information is passed into the Joshua Display window.

C. ERC EXPERT SYSTEM SESSION

1. Selecting a TOE

A TOE is selected using the command *Select TOE* from the command menu. The LISP function **select-toe** is invoked to select the appropriate TOE. All TDS TOE documents must be placed in a physical pathname directory **toe-data** under the logical pathname used for the system. The LISP function looks into this directory to find and display all TOEs available for selection. Figure 5.3 presents the TOE selection process. A possible modification to this operation is to allow a selection of multiple TOE's at one time as compared to only one at a time in this system. Once selected a TOE the ERC EXPERT SYSTEM is now ready to read through the TOE to determine the ERC.

2. Auto Mode Versus Step Mode

The preferred method of running a session is to use the command *Auto Read Current Toe*. This will initiate the reading of the TOE line by line and placing predications into the database for reasoning. No further intervention is required after issuing this command. The alternate command to run a session is *Step Read Current Toe*. This command is specifically used for rule management as discussed in a Section D of this chapter, but can be invoked to effect the side effect of teaching a TOE analyst how the system executes an ERC determination and what rule was used to determine a particular ERC. Both commands are invoked by placing the mouse pointer on the command and clicking the left mouse button. Note all commands are executed by first pointing to the appropriate command and then clicking with the left mouse button. Any

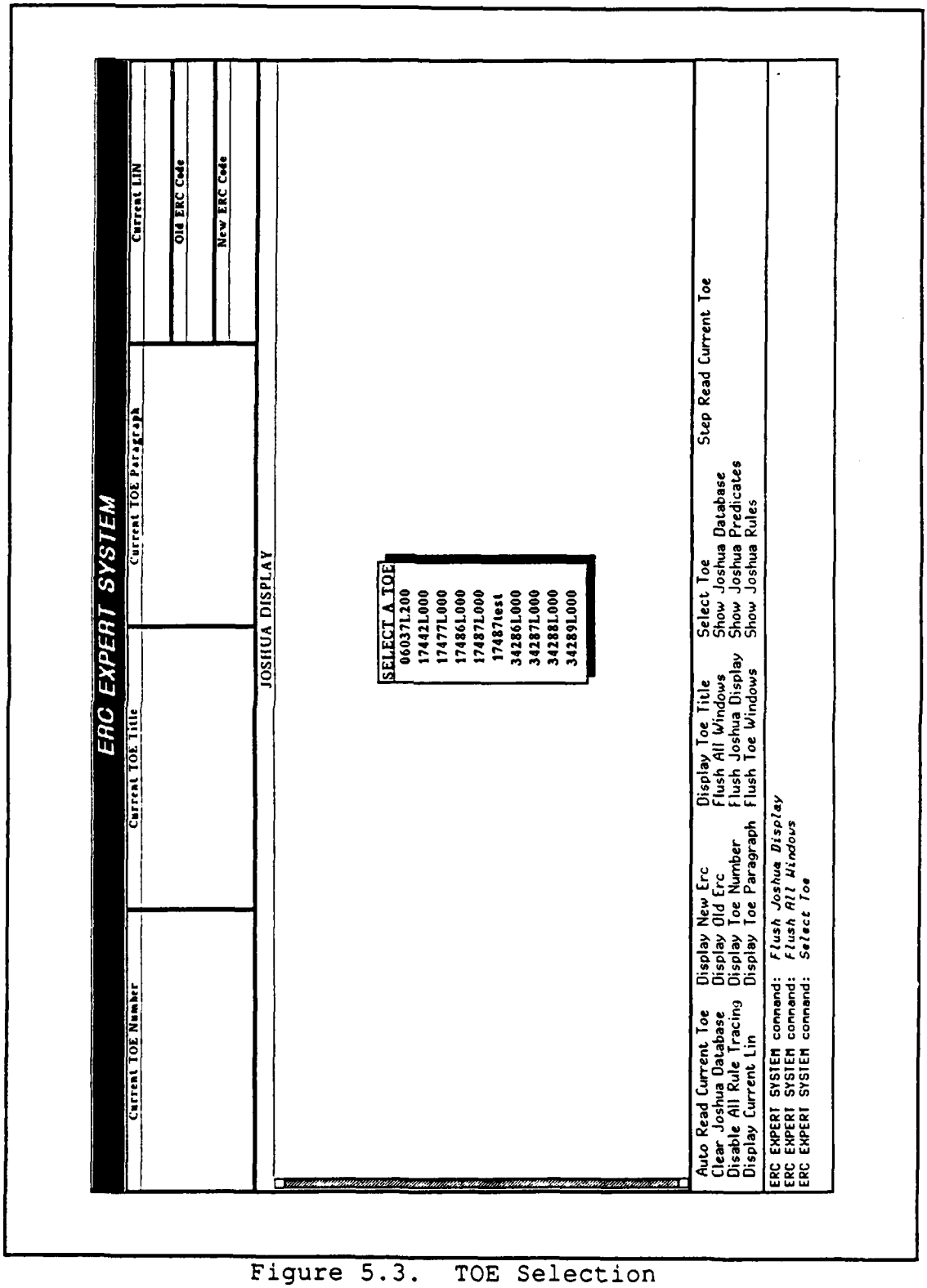

Figure 5.3. TOE Selection

further commands that can be invoked are noted in the standard LISP display at the bottom of the window, noting the appropriate key-mouse combination to issue a command. At the end of a session of a selected TOE, all windows are cleared and a message is displayed to indicate the end of the session. See Figure 5.4 for an example.

3. ERC Determination

The system reads each line in the TOE, skipping through any personnel records, to assert each equipment line, D-record or E-record, into the database as a predication. The unification process matches any knowledge in the database against the system rules to determine an ERC for the current LIN. If so, the rule is triggered and a system message is displayed to the **JOSHUA DISPLAY** window. This procedure is noted in Figure 5.5.

4. Window and Joshua Display Commands

To invoke a specific window or Joshua command is only a matter of "mousing" on the appropriate command in the command menu. The TOE windows, the main display window, or all windows can be flushed at any time. To redisplay any item in the TOE display windows, each TOE window item must be selected individually. Display in the **Joshua Display** occurs again automatically when the next TOE item is read by the system and thus a specific command is not provided.

Three Joshua commands are provided for a user to see the status of the ERC EXPERT SYSTEM. The first, *Show Joshua Database*, displays all current predications in the system under the "belief" of their truth values as concluded by the LTMS. *Show*

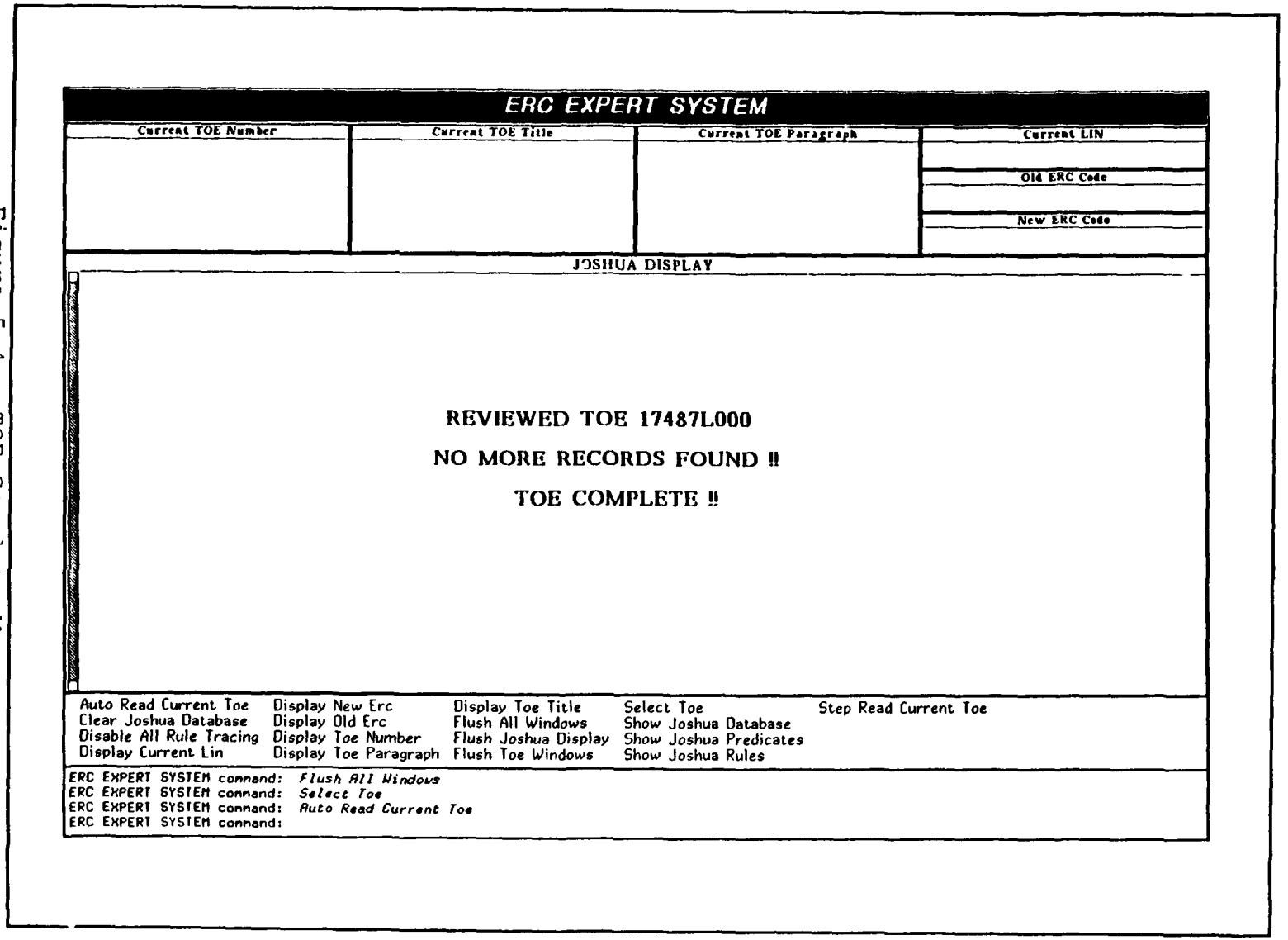

Figure 5.4. TOE Complete Message

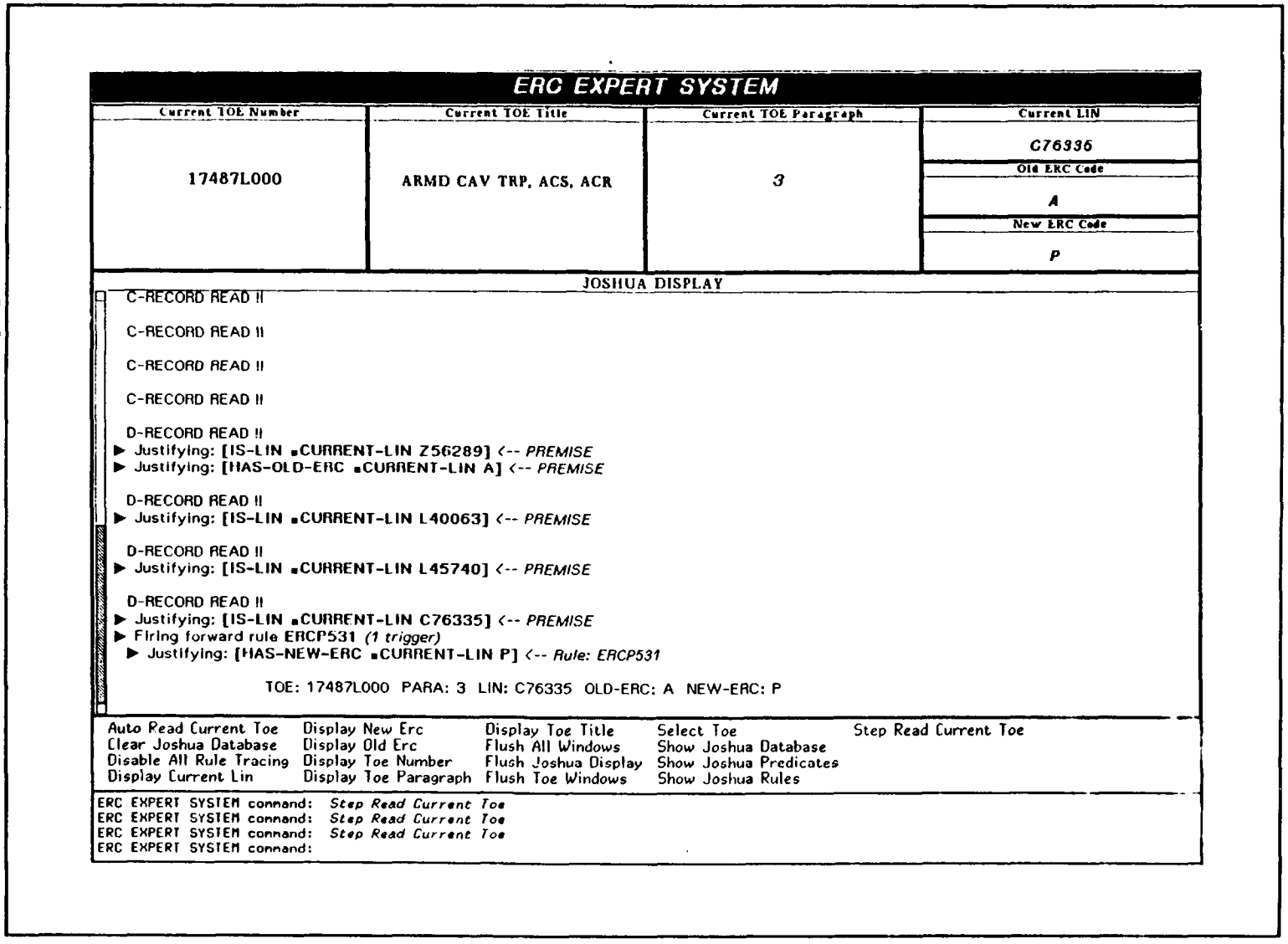

Figure 5.5. Rule Triggering Display

Joshua Predicates presents all predicates definitions. The last Joshua command, *Show Joshua Rules*, offers all currently defined rules in the system. Figure 5.6 outlines all of these commands in one display. Management of these rules is the cornerstone of managing the entire ERC EXPERT SYSTEM process. The following section covers the rule management.

D. ERC RULE MANAGEMENT

One of the most advantageous facets of using of the Joshua software, along with the TMS, is the versatility of rule management. Joshua provides immediate access into the Zmacs editor for rule revision. To invoke this aspect, first move the mouse pointer to any rule name in the **Joshua Display**, whether the name was displayed after a rule trigger or through the use of **Show Joshua Rules** command, and press the **meta** key and the left mouse button simultaneously. This invokes the Zmacs editor buffer and inserts the rule into this buffer for editing. Once editing is complete, compiling the new rule definition and reloading the rule must follow for the database to be updated. If the user desires only to examine the rule definition, moving the mouse pointer to a rule name and clicking the middle mouse button will display the rule definition in the **Joshua Display** window. Figure 5.7 is the invocation of the middle mouse button; Figure 5.8 shows a rule in the Z-Macs buffer after utilizing the **meta** key with the left mouse button. These two mouse features are the foundation for managing the system rules.

The naming convention used in this study is explicit to three items: first, the category under which the rule is found in Appendix I of the CAA study report; secondly,

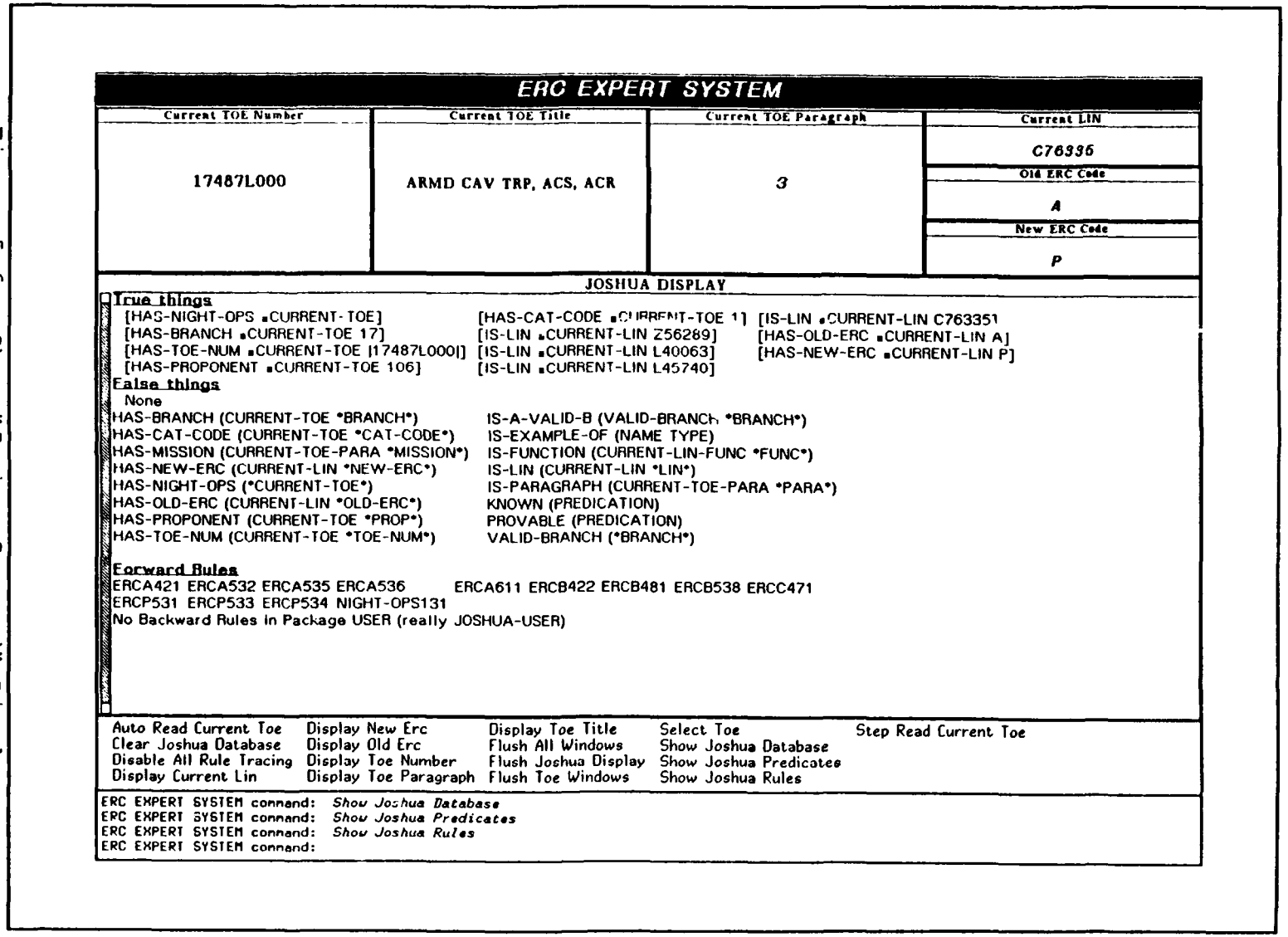

Figure 5.6. Show "Joshua Command" Displays

the number of that category; and finally, the ERC code concluded by the rule. As an example, the name of a rule is generated using a rule found in the CAA study report. The selected rule determines an ERC for a weapon or weapon system that is also a pacing item[Ref. 3:p. I-81]. This rule is found under category **5.3 WEAPON & ASSOCIATED EQUIPMENT RULES**; it is also the first rule in that category and concludes that the ERC of P should be assigned. Thus, a suitable name for this rule is found to be ***ERCP531***. Following the Joshua convention of defining a rule, the rule definition is *(defrule ERCP531 (:forward)* ..., and so forth to complete the entire rule definition. All rules for a specific category are placed in the same LISP file. The LISP rule file corresponds to the category name. In this case, the LISP file maintaining the rules for weapons is **ERC-RULES-WEAPONS.**

E. SUMMARY

Chapter 5 covered the visual interface, conduction of a ERC EXPERT SYSTEM session and most importantly, the management of the rules in the system. The interface provides a user with the necessary information to monitor the ERC determination for equipment within a TOE, insuring proper interpretation of the ERC regulation, AR 220-1. If the current interpretation of a rule is inappropriate or the regulation is changed, the rule management provided by the Joshua software eases the burden of this task by the rule manager.

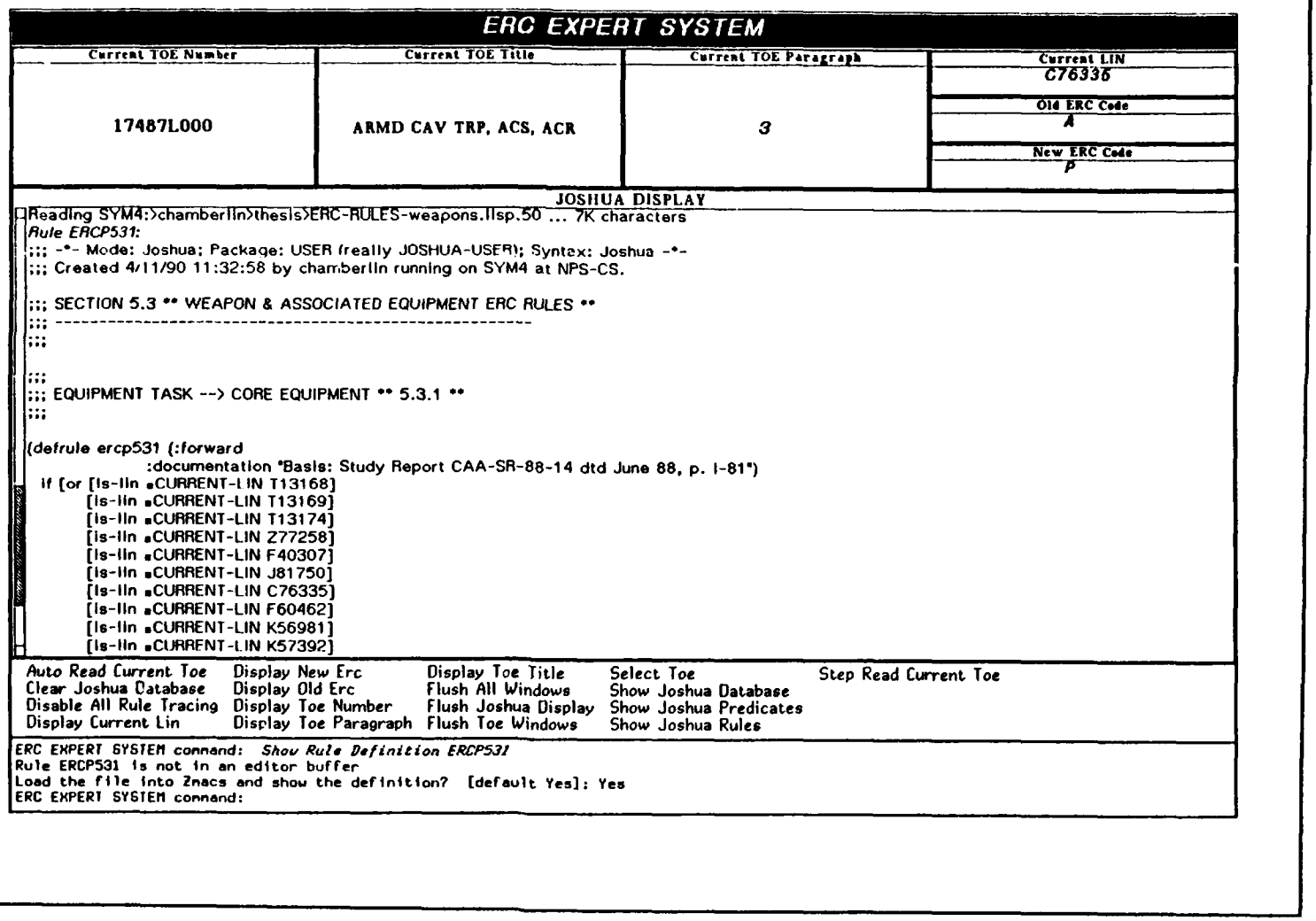

Figure 5.7. Rule Definition Displayed Using Middle Button

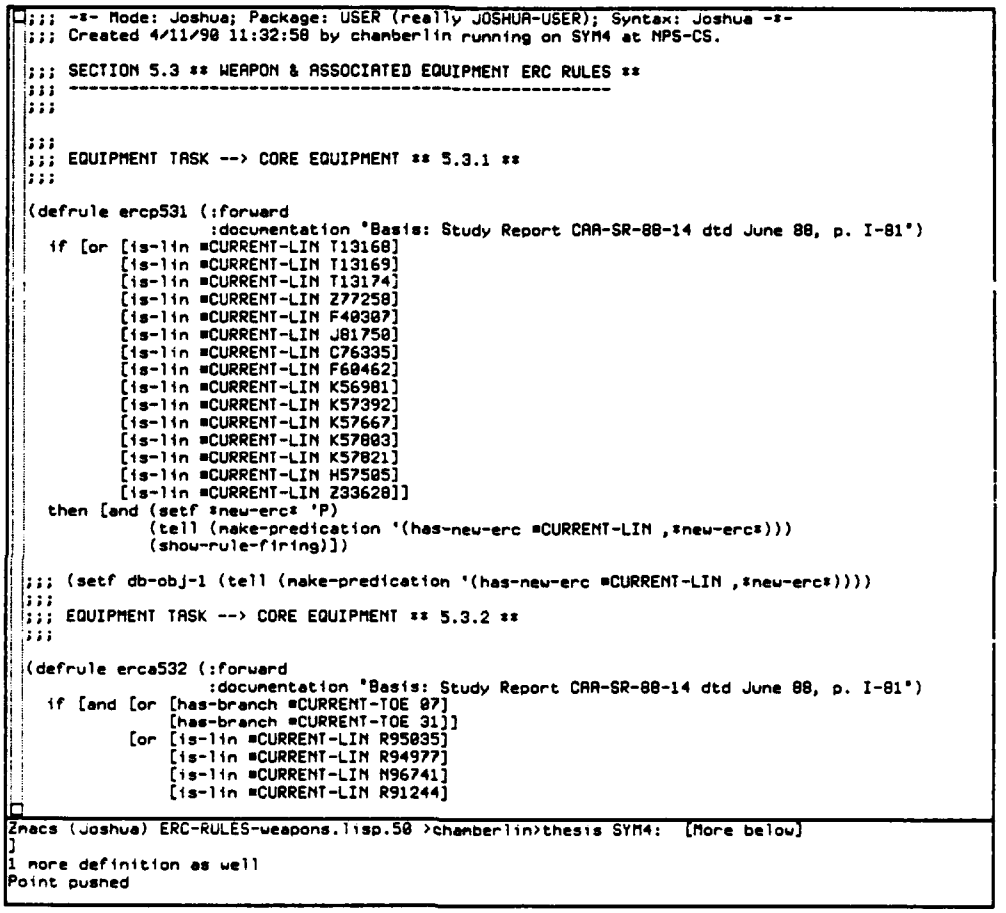

Figure 5.8. Rule in Z-macs Buffer Using **Meta**-Left Button

VI. SUMMARY AND CONCLUSIONS

A. RESEARCH CONTRIBUTIONS

The ERC EXPERT SYSTEM has contributions in the areas of the TOE documentation process, and the development of a complete expert system. The contributions are as follows. First, the system satisfies a directive given by the DCSOPS to produce a rule-based system to handle the current arduous task of determining ERC codes. The task is now simplified, concise and consistent with the rules defined in the system. In addition, the task is no longer a burden to the TOE analyst; the analyst provides the underlying reasoning for the code; the machine produces the ERC code; The second contribution is the ease of integration of this system into concurrent system development at HQ TRADOC. The demonstration of the Joshua software as a builder of expert systems is a third benefit of this study. This software provides excellent tools and features to produce expert system applications. The most important of these are the embedding of LISP code within all Joshua structures, the ability to modify the underlying constructs of the Joshua system to fit application needs, and the integration of rule management directly into the system editor. The final contribution of this study is the application of the *Define-Program-Framework* Flavor in producing the interface. This feature of Symbolics Common LISP has never been attempted at this institution. This Flavor is clearly a step above current windowing structures used in Common Lisp,

particularly since an interactive environment is also provided to generate the basic windows of the interface.

B. RESEARCH EXTENSIONS

It is quite apparent that this study requires some immediate extensions. First, it is necessary to complete all rules as noted in Appendix I of the CAA Study Report. Completion of these rules will make the ERC EXPERT SYSTEM a complete system, primed to automatically generate and/or verify the ERC within all TOE documents. The system can also be extended to be more intelligent of the TOE documents themselves. For example, careful review of the TOE documents will show functionality among groups of personnel and equipment. Thus, rules could find the dependencies between pieces of equipment and then apply codes to "equipment groups", therefore not requiring individual reasoning on each piece of equipment. Meta-rules can render control over rules that may be found to produce contradictions within a document. Lastly, as the developer of this system gains more knowledge and experience of with Joshua, modifications can be made to the TMS to provide better justification to the readiness codes.

APPENDIX A

****

TRADOC DOCUMENTATION SYSTEM
A - RECORD

FLD NO.	NAME/DESCRIPTION	LENGTH	POSITION	CLASS
1	SRC - SRC NUMBER	9	1 - 9	CHAR
2	PAR - PARAGRAPH NO.	3	10 - 12	CHAR
3	- FILLER	13	13 - 25	
4	EDAT-YY - EFFECTIVE DATE-YEAR	2	26 - 27	CHAR
5	EDAT-MM - EFFECTIVE DATE-MONTH	2	28 - 29	CHAR
6	EDAT-DD - EFFECTIVE DATE-DAY	2	30 - 31	CHAR
7	- FILLER	6	32 - 37	
8	MIC - MANAGE. INDICATOR CODE	2	38 - 39	CHAR
9	MEI - MASTER ELEMENT INDICATOR	2	40 - 41	CHAR
10	RTYP - RECORD TYPE(ALWAYS 'A')	1	42	CHAR
11	DLA - DATE LAST ACTION(YYMMDD)	6	43 - 48	CHAR
12	PP - PUBLISH/PROCESS CODE	2	49	CHAR
13	- FILLER	30	50 - 79	
14	CTU-YY - CTU YEAR	2	80 - 81	CHAR
15	CTU-MM - CTU MONTH	2	82 - 83	CHAR
16	PROP - PROPONENT CODE	3	84 - 86	CHAR
17	- FILLER	9	87 - 95	
18	AO - ACTION OFFICER CODE	1	96	CHAR
19	ROD - REPL/OBSOL/DEL/REPLACES	1	97	CHAR
20	STAT - STATUS (D,P, or F)	1	98	CHAR
21	CAT - CATEGORY CODE	1	99	CHAR
22	DNB - DIV/NON-DIV/or BOTH	1	100	CHAR
23	PROJ - PROJECT CODE	1	101	CHAR
24	MARC - MARC CODE	4	102 - 105	CHAR
25	HIST - HISTORY	1	106	CHAR
26	TYP DIV - TYPE DIVISION	1	107	CHAR
27	LOC - LOCATION	1	108	CHAR
28	CHG NO. - CHANGE NUMBER (base ??)	2	109 - 110	CHAR
29	- FILLER	5	111 - 115	
30	ROD SRC - REP/OBS/DEL SRC	9	116 - 124	CHAR
31	TITL - SRC TITLE	40	125 - 164	VARCHAR

APPENDIX B

****

TRADOC DOCUMENTATION SYSTEM
B - RECORD

FLD NO.	NAME/DESCRIPTION			LENGTH	POSITION	CLASS
1	SRC	-	SRC NUMBER	9	1 - 9	CHAR
2	PAR	-	PARAGRAPH NUMBER	3	10 - 12	CHAR
3		-	CONSTANT '00'	2	13 - 14	CHAR
4	CELL	-	CELL IDENTIFIER	6	15 - 20	CHAR
5		-	FILLER	11	21 - 31	
6	BOIP	-	BOIP NUMBER	6	32 - 37	CHAR
7	MIC	-	MANAGE. INDICATOR CODE	2	38 - 39	CHAR
8		-	FILLER	2	40 - 41	
9	RTYP	-	RECORD TYPE (ALWAYS 'B')	1	42	CHAR
10	DLA	-	DATE LAST ACTION(YYMMDD)	6	43 - 48	CHAR
11	PP	-	PUBLISH/PROCESS CODE	1	49	CHAR
12	UM1	-	UNIT MULTIPLIER-1	5	50 - 54	INT
13	UM2	-	UNIT MULTIPLIER-2	5	55 - 59	INT
14	UM3	-	UNIT MULTIPLIER-3	5	60 - 64	INT
15			FILLER	28	65 - 92	
16	AUG RMK	-	AUGMENTATION REMARK	3	93 - 95	CHAR
17			FILLER	6	96 - 101	
18	FUNC	-	FUNCTION CODE	4	102 - 105	CHAR
19			FILLER	3	106 - 108	
20	CHG NO.	-	CHANGE NUMBER (base ?)	2	109 - 110	CHAR
21			FILLER	14	111 - 124	
22	TITL	-	PARAGRAPH TITLE	40	125 - 164	VARCHAR

APPENDIX C

****

TRADOC DOCUMENTATION SYSTEM
C - RECORD

FLD NO.	NAME/DESCRIPTION			LENGTH	POSITION	CLASS
1	SRC	-	SRC NUMBER	9	1 - 9	CHAR
2	PAR	-	PARAGRAPH NUMBER	3	10 - 12	CHAR
3	GR	-	GRADE	2	13 - 14	CHAR
4	MOS	-	MIL. OCCUPATIONAL SPECLTY	6	15 - 20	CHAR
5	SDTC	-	STAND. DUTY TITLE CODE	3	21 - 23	CHAR
6	ASI1	-	ADDNL SKILL IDENTIFIER-1	2	24 - 25	CHAR
7	ASI2	-	ADDNL SKILL IDENTIFIER-2	2	26 - 27	CHAR
8	ASI3	-	ADDNL SKILL IDENTIFIER-3	2	28 - 29	CHAR
9	ASI4	-	ADDNL SKILL IDENTIFIER-4	2	30 - 31	CHAR
10		-	FILLER	6	32 - 37	
11	MIC	-	MGT. INDICATOR CODE	2	38 - 39	CHAR
12	OE	-	OPERATIONAL ELEMENT	2	40 - 41	CHAR
13	RTYP	-	RECORD TYPE (ALWAYS 'C')	1	42	CHAR
14	DLA	-	DATE LAST ACTION (YYMMDD)	6	43 - 48	CHAR
15	PP	-	PUBLISH/PROCESS CODE	1	49	CHAR
16	LVL1	-	STRENGTH LEVEL-1	5	50 - 54	INT
17	LVL2	-	STRENGTH LEVEL-2	5	55 - 59	INT
18	LVL3	-	STRENGTH LEVEL-3	5	60 - 64	INT
19	LVLA	-	STRENGTH LEVEL-A	5	65 - 69	INT
20	LVLB	-	STRENGTH LEVEL-B	5	70 - 74	INT
21	LVLC	-	STRENGTH LEVEL-C	5	75 - 79	INT
22	BR	-	BRANCH	2	80 - 81	CHAR
23	DCPC	-	DIRECT COMBAT PROBLTY CODE	2	82 - 83	CHAR
24	NOTE1	-	NOTE-1	3	84 - 86	CHAR
25	NOTE2	-	NOTE-2	3	87 - 89	CHAR
26	NOTE3	-	NOTE-3	3	90 - 92	CHAR
27			FILLER	16	93 - 108	
28	CHG NO.	-	CHANGE NUMBER (base ?)	2	109 - 110	CHAR
29			FILLER	54	111 - 164	

APPENDIX D

****

TRADOC DOCUMENTATION SYSTEM
D - RECORD

FLD NO.	NAME/DESCRIPTION	LENGTH	POSITION	CLASS
1	SRC - SRC NUMBER	9	1 - 9	CHAR
2	PAR - PARAGRAPH NUMBER	3	10 - 12	CHAR
3	- FILLER	2	13 - 14	
4	LIN - LINE ITEM NUMBER	6	15 - 20	CHAR
5	ERC - EQP. READINESS CODE	3	21 - 23	CHAR
6	- FILLER	8	24 - 31	
7	BOIP - BASIS OF ISSUE PLAN NBR	6	32 - 37	CHAR
8	MIC - MGT. INDICATOR CODE	2	38 - 39	CHAR
9	OE - OPERATIONAL ELEMENT	2	40 - 41	CHAR
10	RTYP - RECORD TYPE (ALWAYS 'D')	1	42	CHAR
11	DLA - DATE LAST ACTION (YYMMDD)	6	43 - 48	CHAR
12	PP - PUBLISH/PROCESS CODE	1	49	CHAR
13	LVL1 - STRENGTH LEVEL-1	5	50 - 54	CHAR
14	LVL2 - STRENGTH LEVEL-2	5	55 - 59	CHAR
15	LVL3 - STRENGTH LEVEL-3	5	60 - 64	CHAR
16	LVLA - STRENGTH LEVEL-A	5	65 - 69	CHAR
17	LVLB - STRENGTH LEVEL-B	5	70 - 74	CHAR
18	- FILLER	9	75 - 83	CHAR
20	NOTE1 - NOTE-1	3	84 - 86	CHAR
21	NOTE2 - NOTE-2	3	87 - 89	CHAR
22	NOTE3 - NOTE-3	3	90 - 92	CHAR
23	EQP RMK - EQUIPMENT REMARK	3	93 - 95	CHAR
24	DTOE - ?	1	96	CHAR
25	QTY ADJ - QUANTITY ADJUSTMENT CODE	1	97	CHAR
26	APP TOE - BOIP APPLIED TO TOE CODE	1	98	CHAR
27	OSR - OTHER SUPPORT REQUIREMENT	1	99	CHAR
28	- FILLER	9	100 - 108	
29	CHG NBR - CHANGE NUMBER (Old base)	2	109 - 110	CHAR
30	- FILLER	54	111 - 164	

APPENDIX E

****

TRADOC DOCUMENTATION SYSTEM
E - RECORD

FLD NO.	NAME/DESCRIPTION			LENGTH	POSITION	CLASS
1	SRC	-	SRC NUMBER	9	1 - 9	CHAR
2	PAR	-	PARAGRAPH NUMBER	3	10 - 12	CHAR
3		-	FILLER	2	13 - 14	
4	LIN	-	LINE ITEM NUMBER	6	15 - 20	CHAR
5	ERC	-	EQUIPMENT READINESS CODE	3	21 - 23	CHAR
6		-	FILLER	14	24 - 37	
7	MIC	-	MGT. INDICATOR CODE	2	38 - 39	CHAR
8	OE	-	OPERATIONAL ELEMENT	2	40 - 41	CHAR
9	RTYP	-	RECORD TYPE (ALWAYS 'E')	1	42	CHAR
10	DLA	-	DATE LAST ACTION(YYMMDD)	6	43 - 48	CHAR
11	PP	-	PUBLISH/PROCESS CODE	1	49	CHAR
12	LVL1	-	STRENGTH LEVEL-1	5	50 - 54	INT
13	LVL2	-	STRENGTH LEVEL-2	5	55 - 59	INT
14	LVL3	-	STRENGTH LEVEL-3	5	60 - 64	INT
15	LVLA	-	STRENGTH LEVEL-A	5	65 - 69	INT
16	LVLB	-	STRENGTH LEVEL-B	5	70 - 74	INT
17		-	FILLER	9	75 - 83	
18	NOTE1	-	NOTE NUMBER 1	3	84 - 86	CHAR
19	NOTE2	-	NOTE NUMBER 2	3	87 - 89	CHAR
20	NOTE3	-	NOTE NUMBER 3	3	90 - 92	CHAR
21	EQP RMK	-	EQUIPMENT REMARK NUMBER	3	93 - 95	CHAR
22		-	FILLER	13	96 - 108	
23	CHG NBR	-	CHANGE NUMBER (Old Base)	2	109 - 110	CHAR
24		-	FILLER	54	111 - 164	

APPENDIX F

****

TRADOC DOCUMENTATION SYSTEM
F - RECORD

FLD NO.	NAME/DESCRIPTION			LENGTH	POSITION	CLASS
1	SRC	-	SRC NUMBER	9	1 - 9	CHAR
2	PAR	-	PARAGRAPH NUMBER	3	10 - 12	CHAR
3		-	FILLER	2	13 - 14	
4	FRMK	-	F-RECORD REMARK NUMBER	6	15 - 20	CHAR
5	SEQ	-	REMARK SEQUENCE	3	21 - 23	CHAR
6		-	FILLER	14	24 - 37	CHAR
7	MIC	-	MANAGE. INDICATOR CODE	2	38 - 39	CHAR
8		-	FILLER	2	40 - 41	
9	RTYP	-	RECORD TYPE (ALWAYS 'F')	1	42	CHAR
10	DLA	-	DATE LAST ACTION(YYMMDD)	6	43 - 48	CHAR
11	PP	-	PUBLISH/PROCESS CODE	1	49	CHAR
12		-	FILLER	59	50 - 108	
13	CHG NBR	-	CHANGE NUMBER (Old Base)	2	109 - 110	CHAR
14		-	FILLER	14	111 - 124	
15	RMK LINE	-	REMARK	40	125 - 164	CHAR

APPENDIX G

TRADOC DOCUMENTATION SYSTEM
P - RECORD

FLD NO.	NAME/DESCRIPTION			LENGTH	POSITION	CLASS
1	SRC	-	SRC NUMBER	9	1 - 9	CHAR
2	PAR	-	PARAGRAPH NUMBER	3	10 - 12	CHAR
3	GR	-	GRADE	2	13 - 14	CHAR
4	MOS	-	MIL OCCUPATIONAL SKILL	6	15 - 20	CHAR
5	SDTC	-	STD. DUTY TITLE CODE NBR	3	21 - 23	
6	ASI1	-	ADDNL SKILL IDENT.-1	2	24 - 25	CHAR
7	ASI2	-	ADDNL SKILL IDENT.-2	2	26 - 27	CHAR
8	ASI3	-	ADDNL SKILL IDENT.-3	2	28 - 29	CHAR
9	ASI4	-	ADDNL SKILL IDENT.-4	2	30 - 31	CHAR
10	BOIP	-	BASIS OF ISSUE PLAN NBR	6	32 - 37	CHAR
11	MIC	-	MGT. INDICATOR CODE	2	38 - 39	CHAR
12	OE	-	OPERATIONAL ELEMENT	2	40 - 41	CHAR
13	RTYP	-	RECORD TYPE (ALWAYS 'P')	1	42	CHAR
14	DLA	-	DATE LAST ACTION (YYMMDD)	6	43 - 48	CHAR
15	PP	-	PUBLISH/PROCESS CODE	1	49	CHAR
16	LVL1	-	STRENGTH LEVEL-1	5	50 - 54	INT
17	LVL2	-	STRENGTH LEVEL-2	5	55 - 59	INT
18	LVL3	-	STRENGTH LEVEL-3	5	60 - 64	INT
19	LVLA	-	STRENGTH LEVEL-A	5	65 - 69	INT
20	LVLB	-	STRENGTH LEVEL-B	5	70 - 74	INT
21	LVLC	-	STRENGTH LEVEL-C	5	75 - 79	INT
22	BR	-	BRANCH	2	80 - 81	CHAR
23	DCPC	-	DIRECT COMBAT PROB. CODE	2	82 - 83	CHAR
24	NOTE1	-	NOTE NUMBER-1	3	84 - 86	CHAR
25	NOTE2	-	NOTE NUMBER-2	3	87 - 89	CHAR
26	NOTE3	-	NOTE NUMBER 3	3	90 - 92	CHAR
27		-	FILLER	3	93 - 95	
28	DTOE	-	?	1	96	CHAR
29	QTY ADJ	-	QUANTITY ADJUSTMENT CODE	1	97	CHAR
30	APP TOE	-	BOIP APPLIED TO TOE CODE	1	98	CHAR
31	OSR	-	ORGANIZATIONAL SPT. RQMTS.	1	99	CHAR
32		-	FILLER	9	100 - 108	
33	CHG NBR	-	CHANGE NUMBER (Old Base)	2	109 - 110	CHAR
34		-	FILLER	54	111 - 164	

APPENDIX H

```
;;; -*- Mode: Joshua; Package: JOSHUA-USER; Syntax: Joshua -*-
;;; Created 4/05/90 10:39:58 by chamberlin running on SYM4 at NPS-CS.


;*****************************************************************************************
;
;   FILENAME.......... :   erc-auto-read-toe.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   5 Apr 90
;   FILE DESCRIPTION... :   Contains the methods invoked by the "Auto Read Current TOE"
;                           on the Interface command menu.  A single TOE record is read
;                           tested for record type, and then depending on the record
;                           instantiates global variables or is skipped.
;
;   MODIFICATIONS..... :   10 May 90 - Adjusted format of messages
;
;*****************************************************************************************


(defun auto-read-current-toe ()
  (setf *toe-record* (setf *end-of-file* (read-line *in-file* nil)))
  (cond ((not (equalp *toe-record* nil))
         (auto-test-for-record))
        (t (auto-read-current-toe))))

(defun auto-test-for-record ()
  (cond ((auto-test-for-A-rec) (format t "~%~@3TA-RECORD READ !!~%")
                               (setf *title-length* (- (string-length *toe-record*) 124))
                               (get-toe-data *toe-record*)
                               (auto-read-current-toe))
        ((auto-test-for-B-rec) (format t "~%~@3TB-RECORD READ !!~%")
                               (get-toe-para *toe-record*)
                               (auto-read-current-toe))
        ((auto-test-for-C-rec) (format t "~%~@3TC-RECORD READ !!~%")
                               (auto-read-current-toe))
        ((auto-test-for-D-rec) (format t "~%~@3TD-RECORD READ !!~%")
                               (get-equip-data *toe-record*)
                               (auto-read-current-toe))
        ((auto-test-for-E-rec) (format t "~%~@3TE-RECORD READ !!~%")
                               (get-equip-data *toe-record*)
                               (auto-read-current-toe))
        ((auto-test-for-F-rec) (format t "~%~@3TF-RECORD READ !!~%")
                               (auto-read-current-toe))
        ((auto-test-for-P-rec) (format t "~%~@3TP-RECORD READ !!~%")
                               (auto-read-current-toe))
        (t (com-flush-joshua-display)
           (format t "~8%~@59T~vREVIEWED TOE ~A~%
                   '(:dutch :bold :very-large) *selected-toe*)
           (format t "~2%~@57T~vNO MORE RECORDS FOUND !!~%
                   '(:dutch :bold :very-large))
           (format t "~2%~@70T~vTOE COMPLETE !!~%
                   '(:dutch :bold :very-large)))))
```

```
(defun auto-test-for-A-rec ()
  (if (eq (string-search-char '#\A *toe-record* :start 41) 41) t))

(defun auto-test-for-B-rec ()
  (if (eq (string-search-char '#\B *toe-record* :start 41) 41) t))

(defun auto-test-for-C-rec ()
  (if (eq (string-search-char '#\C *toe-record* :start 41) 41) t))

(defun auto-test-for-D-rec ()
  (if (eq (string-search-char '#\D *toe-record* :start 41) 41) t))

(defun auto-test-for-E-rec ()
  (if (eq (string-search-char '#\E *toe-record* :start 41) 41) t))

(defun auto-test-for-F-rec ()
  (if (eq (string-search-char '#\F *toe-record* :start 41) 41) t))

(defun auto-test-for-P-rec ()
  (if (eq (string-search-char '#\P *toe-record* :start 41) 41) t))
```

APPENDIX I

```
;;; -*- Mode: Joshua; Package: JOSHUA-USER; Syntax: Joshua -*-
;;; Created 5/07/90 11:17:59 by chamberlin running on SYM4 at NPS-CS.

;*****************************************************************************
;
;   FILENAME.......... :   erc-functions.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   7 May 90
;   FILE DESCRIPTION... :   Contains the all methods used by the entire system to
;                           instantiate all the global variables, which in turn are
;                           used as arguments to the predicate definitions.  The
;                           global variables are defined by the record type and the
;                           TDS record layouts (Appendices A-G).
;
;   MODIFICATIONS..... :   12 May 90 - added 'merge pathnames' for function
;                                      'select-new-toe'
;
;*****************************************************************************


(defun select-new-toe ()
  (setf *selected-toe* nil)
  (select-toe)
  (setf *in-file* (open (merge-pathnames *selected-toe* "sym4:>chamberlin>toe-data>.dat"))))


(defun get-toe-data (*toe-record*)
  (setf *toe-num*
        (intern (string-append (string (aref *toe-record* 0))
                               (string (aref *toe-record* 1))
                               (string (aref *toe-record* 2))
                               (string (aref *toe-record* 3))
                               (string (aref *toe-record* 4))
                               (string (aref *toe-record* 5))
                               (string (aref *toe-record* 6))
                               (string (aref *toe-record* 7))
                               (string (aref *toe-record* 8)))))
  (setf *toe-title* "")
  (do ((counter 0 (- counter 1)))
      ((eq counter *title-length*) *toe-title*)
    (setf *title-char*
          (string (aref *toe-record* (- 124 counter))))
    (setf *toe-title* (string-append *toe-title* *title-char*)))

  (setf *branch* (parse-integer *toe-record* :start 0 :end 2))
  (setf *prop* (parse-integer *toe-record* :start 83 :end 86))
  (setf *cat-code* (parse-integer *toe-record* :start 98 :end 99))

  (tell (make-predication '(has-toe-num =CURRENT-TOE ,*toe-num*)))
  (tell (make-predication '(has-branch =CURRENT-TOE ,*branch*)))
  (tell (make-predication '(has-proponent =CURRENT-TOE ,*prop*)))
  (tell (make-predication '(has-cat-code =CURRENT-TOE ,*cat-code*))))
```

```
(defun get-toe-para (*toe-record*)
  (setf *para* (parse-integer *toe-record* :start 9 :end 11)))


(defun get-equip-data (*toe-record*)
  (setf *lin*
        (intern (string-append (string (aref *toe-record* 14))
                               (string (aref *toe-record* 15))
                               (string (aref *toe-record* 16))
                               (string (aref *toe-record* 17))
                               (string (aref *toe-record* 18))
                               (string (aref *toe-record* 19))))))
  (setf *old-erc*
        (intern (string-append (string (aref *toe-record* 20)))))
  (tell (make-predication '(is-lin ■CURRENT-LIN ,*lin*)))
  (tell (make-predication '(has-old-erc ■CURRENT-LIN ,*old-erc*))))


(defun show-rule-firing ()
  (format t "~%~25@TTOE: ~A  PARA: ~A  LIN: ~A  OLD-ERC: ~A  NEW-ERC: ~A~%"
          *toe-num* *para* *lin* *old-erc* *new-erc*))
```

APPENDIX J

```
;;; -*- Mode: Joshua; Package: USER (really JOSHUA-USER); Syntax: Joshua -*-
;;; Created 4/05/90 10:52:48 by cnamberlin running on SYM4 at NPS-CS.

;*****************************************************************************************
;
;   FILENAME.......... :   erc-initial-data.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   5 April 90
;   FILE DESCRIPTION... :   Contains all of the global variables for the system.
;                           Also instantiates any variables prior to system run.
;
;
;
;   MODIFICATIONS...... :
;
;
;*****************************************************************************************

;;;
;;; DEFINE ALL GLOBAL VARIABLES
;;;

(defvar *lin*)              ;;; Line Item Number, read from TOE D & E records
(defvar *new-erc*)          ;;; ERC assigned to LIN by a rule
(defvar *old-erc*)          ;;; ERC that is currently in TOE document
(defvar *toe-num*)          ;;; TOE number, read from A record
(defvar *toe-record*)       ;;;
(defvar *toe-title*)        ;;; TOE title, read from A record
(defvar *title-char*)       ;;; A single character of TOE title, used to determine title length
(defvar *title-length*)     ;;; Length of TOE title
(defvar *branch*)           ;;; Branch of the TOE
(defvar *func*)             ;;; Function of TOE paragraph or LIN
(defvar *mission*)          ;;; Mission of TOE or TOE paragraph
(defvar *prop*)             ;;; Proponent of the TOE, read from A record
(defvar *cat-code*)         ;;; Category code of TOE, read from A record
(defvar *para*)             ;;; Paragraph of TOE, read from B record
(defvar *in-file*)          ;;; The variable used to capture which TOE to use
(defvar *end-of-file*)      ;;; End of file marker
(defvar *selected-toe*)     ;;; TOE chosen to run thru system

;;;
;;; INITIALIZE VARIABLES
;;;

(setf *new-erc* 'unknown)
```

APPENDIX K

```
;;; -*- Mode: Joshua; Package: USER (really JOSHUA-USER); Syntax: Joshua -*-
;;; Created 4/23/90 12:54:42 by chamberlin running on SYM1 at NPS-CS.
;*************************************************************** ************************
;
;   FILENAME.......... :   erc-interface.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   23 Apr 90
;   FILE DESCRIPTION... :   This generates the interface windows for the expert system
;                           Key items to note here are:
;                            1. create desired command needed on Command Menu first
;                            2. delete current command table - cp:delete-command-table
;                            3. compile entire file to update commands
;                            4. compile 'define-program-framework' region to recreate
;                               window interface
;
;
;   MODIFICATIONS..... :   18 May 90 - updated display messages
;
;
;*************************************************************************************************

(DW:DEFINE-PROGRAM-FRAMEWORK ERC-EXPERT-SYSTEM
  :pretty-name "ERC EXPERT SYSTEM"
  :SELECT-KEY
  #\Circle
  :COMMAND-DEFINER
  T
  :COMMAND-TABLE
  (:INHERIT-FROM '("colon full command" "standard arguments" "input editor compatibility"
                   "global" "user")
   :KBD-ACCELERATOR-P 'T)
  :STATE-VARIABLES
  nil
  :terminal-io-pane JOSHUA
  :selected-pane JOSHUA
  :PANES
  ((ERC-EXPERT-SYSTEM :TITLE
                      :HEIGHT-IN-LINES 1
                      :reverse-video-p t
                      :REDISPLAY-AFTER-COMMANDS t
                      :default-character-style '(:sans-serif :bold-italic :very-large))
   (TOE-NUM :DISPLAY
            :default-character-style '(:swiss :roman :normal)
            :redisplay-after-commands t
            :more-p t
            :margin-components '((dw:margin-borders :thickness 2)
                                 (dw:margin-label :margin :top
                                                  :box :inside
                                                  :centered-p t
                                                  :style (:dutch :bold :small)
                                                  :string "Current TOE Number")))
   (TOE-TITLE :DISPLAY
              :default-character-style '(:swiss :roman :normal)
              :redisplay-after-commands t
              :more-p nil
              :margin-components '((dw:margin-borders :thickness 2)
                                   (dw:margin-label :margin :top
                                                    :box :inside
                                                    :centered-p t
                                                    :style (:dutch :bold :small)
                                                    :string "Current TOE Title")))
```

```
(TOE-PARA :DISPLAY
          :default-character-style '(:swiss :roman :normal)
          :redisplay-after-commands t
          :more-p nil
          :margin-components '((dw:margin-borders :thickness 2)
                               (dw:margin-label :margin :top
                                                :box :inside
                                                :centered-p t
                                                :style (:dutch :bold :small)
                                                :string "Current TOE Paragraph")))
(LIN :DISPLAY
     :default-character-style '(:swiss :roman :normal)
     :redisplay-after-commands t
     :more-p nil
     :margin-components '((dw:margin-borders :thickness 2)
                          (dw:margin-label :margin :top
                                           :box :inside
                                           :centered-p t
                                           :style (:dutch :bold :small)
                                           :string "Current LIN")))
(OLD-ERC :DISPLAY
         :default-character-style '(:swiss :roman :normal)
         :redisplay-after-commands t
         :more-p nil
         :margin-components '((dw:margin-borders :thickness 2)
                              (dw:margin-label :margin :top
                                               :box :inside
                                               :centered-p t
                                               :style (:dutch :bold :small)
                                               :string "Old ERC Code")))
(NEW-ERC :DISPLAY
         :default-character-style '(:swiss :roman :normal)
         :redisplay-after-commands t
         :more-p nil
         :margin-components '((dw:margin-borders :thickness 2)
                              (dw:margin-label :margin :top
                                               :box :inside
                                               :centered-p t
                                               :style (:dutch :bold :small)
                                               :string "New ERC Code")))
(JOSHUA :DISPLAY
     :HEIGHT-IN-LINES 25
     :default-character-style '(:swiss :roman :normal)
     :redisplay-after-commands nil
     :more-p t
      margin-components '((dw:margin-borders :thickness 2)
                          (dw:margin-scroll-bar :margin :left)
                          (dw:margin-label :margin :top
                                           :box :inside
                                           :centered-p t
                                           :style (:dutch :bold :normal)
                                           :string "JOSHUA DISPLAY")))
```

```
    (COMMAND-MENU-1 :COMMAND-MENU
                    :EQUALIZE-COLUMN-WIDTHS nil
                    :CENTER-P nil
                    :ROWS t
                    :MENU-LEVEL :TOP-LEVEL)

    (INTERACTOR-1 :INTERACTOR
                  :HEIGHT-IN-LINES 4))

  :CONFIGURATIONS
  '((DW::MAIN
     (:LAYOUT
      (DW::MAIN :COLUMN ERC-EXPERT-SYSTEM ROW-1 JOSHUA COMMAND-MENU-1 INTERACTOR-1)
      (ROW-1 :ROW TOE-NUM TOE-TITLE TOE-PARA COLUMN-1) (COLUMN-1 :COLUMN LIN OLD-ERC NEW-ERC))
     (:SIZES
      (DW::MAIN (ERC-EXPERT-SYSTEM 1 :LINES) (JOSHUA 25 :LINES)
       (COMMAND-MENU-1 :ASK-WINDOW SELF :SIZE-FOR-PANE COMMAND-MENU-1) (INTERACTOR-1 4 :LINES)
       :THEN (ROW-1 :even))
      (ROW-1 (TOE-NUM :even) (TOE-TITLE :even) (TOE-PARA :even ) (COLUMN-1 :even))
      (COLUMN-1 (LIN :EVEN) (OLD-ERC :EVEN) (NEW-ERC :EVEN))))))


(define-erc-expert-system-command
  (com-display-toe-number :menu-accelerator t)
    ()
   (send (dw:get-program-pane 'TOE-NUM) :clear-history)
   (let ((*standard-output* (dw:get-program-pane 'TOE-NUM)))
     (format t "~3%~15@T~v@a~%
             '(:dutch :bold :large) "toe-num")))

(define-erc-expert-system-command
  (com-display-toe-title :menu-accelerator t)
    ()
   (send (dw:get-program-pane 'TOE-TITLE) :clear-history)
   (let ((*standard-output* (dw:get-program-pane 'TOE-TITLE)))
     (format t "~3%~5@T~v@a~%
             '(:dutch :bold :normal) "toe-title")))

(define-erc-expert-system-command
  (com-display-toe-paragraph :menu-accelerator t)
    ()
   (send (dw:get-program-pane 'TOE-PARA) :clear-history)
   (let ((*standard-output* (dw:get-program-pane 'TOE-PARA)))
     (format t "~3%~20@T~v@a~%
             '(:swiss :bold-italic :large) "para")))

(define-erc-expert-system-command
  (com-display-current-lin :menu-accelerator t)
    ()
   (send (dw:get-program-pane 'LIN) :clear-history)
   (let ((*standard-output* (dw:get-program-pane 'LIN)))
     (format t "~%~17@T~v@a~% '(:swiss :bold-italic :normal) "lin")))

(define-erc-expert-system-command
  (com-display-new-erc :menu-accelerator t)
    ()
   (send (dw:get-program-pane 'NEW-ERC) :clear-history)
   (let ((*standard-output* (dw:get-program-pane 'NEW-ERC)))
     (cond ((eq *new-erc* 'unknown)
            (format t "~%~10@T~v@a~% EVALUATED~% '(:swiss :bold-italic :normal)))
           (t (format t "~%~20@T~v@a~% '(:swiss :bold-italic :normal) *new-erc*)))))
```

```
(define-erc-expert-system-command
  (com-display-old-erc :menu-accelerator t)
    ()
   (send (dw:get-program-pane 'OLD-ERC) :clear-history)
   (let ((*standard-output* (dw:get-program-pane 'OLD-ERC)))
     (format t "~%~20@T~v@a~5 '(:swiss :bold-italic :normal) *old-erc*)))

(define-erc-expert-system-command
  (com-show-joshua-database :menu-accelerator t)
    ()
   (let ((*standard-output* (dw:get-program-pane 'JOSHUA)))
     (ji::com-show-joshua-database)))

(define-erc-expert-system-command
  (com-show-joshua-rules :menu-accelerator t)
    ()
;    ((type 'keyword :default ':Type)
;     (direction 'symbol :default 'forward))
   (let ((*standard-output* (dw:get-program-pane 'JOSHUA)))
     (ji::com-show-joshua-rules)))
;     (ji::com-show-joshua-rules 'type 'direction)))

(define-erc-expert-system-command
  (com-show-joshua-predicates :menu-accelerator t)
    ()
   (let ((*standard-output* (dw:get-program-pane 'JOSHUA)))
     (ji::com-show-joshua-predicates)))

;(define-erc-expert-system-command
;  (com-trace-forward-rules :menu-accelerator t)
;    ()
;   (let ((*standard-output* (dw:get-program-pane 'JOSHUA)))
;     (com-enable-joshua-tracing-forward-rules)))

(define-erc-expert-system-command
  (com-disable-all-rule-tracing :menu-accelerator t)
    ()
   (let ((*standard-output* (dw:get-program-pane 'JOSHUA)))
     (ji::com-disable-joshua-tracing)))

(define-erc-expert-system-command
  (com-flush-joshua-display :menu-accelerator t)
    ()
   (send (dw:get-program-pane 'JOSHUA) :clear-history))

(define-erc-expert-system-command
  (com-flush-TOE-windows :menu-accelerator t)
    ()
   (send (dw:get-program-pane 'TOE-NUM) :clear-history)
   (send (dw:get-program-pane 'TOE-TITLE) :clear-history)
   (send (dw:get-program-pane 'TOE-PARA) :clear-history)
   (send (dw:get-program-pane 'LIN) :clear-history)
   (send (dw:get-program-pane 'OLD-ERC) :clear-history)
   (send (dw:get-program-pane 'NEW-ERC) :clear-history))
```

```
(define-erc-expert-system-command
  (com-select-toe :menu-accelerator t)
    ()
   (com-flush-all-windows)
   (clear)
   (select-new-toe)
   (com-flush-joshua-display)
   (format t
           "~10%~5@T~vCOE ~A IS LOADED AND READY TO RUN THROUGH~%
           '(:dutch :bold :very-large) *selected-toe*)
   (format t
           "~vCERC EXPERT SYSTEM ~%
           '(:swiss :bold-italic :very-large)))

(define-erc-expert-system-command
  (com-step-read-current-toe :menu-accelerator t)
    ()
   (let ((*standard-output* (dw:get-program-pane 'JOSHUA))))
   (read-current-toe)
   (cond ((test-for-A-rec)
          (com-display-toe-number)
          (com-display-toe-title))
         ((test-for-B-rec)
          (com-display-toe-paragraph))
         ((test-for-D-rec)
          (com-display-current-lin)
          (com-display-old-erc)
          (com-display-new-erc))
         ((test-for-E-rec)
          (com-display-current-lin)
          (com-display-old-erc)
          (com-display-new-erc)))
   (setf *new-erc* 'unknown))

(define-erc-expert-system-command
  (com-auto-read-current-toe :menu-accelerator t)
    ()
   (com-flush-all-windows)
   (send (dw:get-program-pane 'JOSHUA) :set-more-p nil)
   (auto-read-current-toe)
   (send (dw:get-program-pane 'JOSHUA) :set-more-p t))


(define-erc-expert-system-command
  (com-flush-all-windows :menu-accelerator t)
    ()
   (send (dw:get-program-pane 'JOSHUA) :clear-history)
   (send (dw:get-program-pane 'TOE-NUM) :clear-history)
   (send (dw:get-program-pane 'TOE-TITLE) :clear-history)
   (send (dw:get-program-pane 'TOE-PARA) :clear-history)
   (send (dw:get-program-pane 'LIN) :clear-history)
   (send (dw:get-program-pane 'OLD-ERC) :clear-history)
   (send (dw:get-program-pane 'NEW-ERC) :clear-history))

(define-erc-expert-system-command
  (com-clear-joshua-database :menu-accelerator t)
    ()
   (com-flush-all-windows)
   (clear)
   (ji::com-show-joshua-database))

;(define-erc-expert-system-command
;  (com-untell :menu-accelerator t)
;    ()
```

APPENDIX L

```
;;; -*- Mode: Joshua; Package: USER (really JOSHUA-USER); Syntax: Joshua -*-
;;; Created 4/12/90 14:19:30 by chamberlin running on SYM4 at NPS-CS.

;*****************************************************************************************
;
;   FILENAME.......... :   erc-predicates.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   12 Apr 90
;   FILE DESCRIPTION... :   Contains predicates definitions or the ERC EXPERT SYSTEM
;
;   MODIFICATIONS..... :
;
;*****************************************************************************************

;;; MISSION PREDICATES
;;;

(define-predicate has-night-ops (*current-toe*)
                  (ltms:ltms-predicate-model))  ;;; identifies if a TOE requires night ops

;;; TOE SPECIFIC PREDICATES


(define-predicate is-paragraph (current-toe-para *para*)
                  (ltms:ltms-predicate-model))  ;;; identifies current TOE paragraph

(define-predicate is-a-valid-b (valid-branch *branch*)
                  (ltms:ltms-predicate-model))  ;;; ensure branch read from TOE is valid

(define-predicate has-branch (current-toe *branch*)
                  (ltms:ltms-predicate-model))  ;;; identifies current TOE branch

(define-predicate has-toe-num (current-toe *toe-num*)
                  (ltms:ltms-predicate-model))  ;;; identifies current TOE number

(define-predicate has-proponent (current-toe *prop*) ;;; identifies proponent for current TOE
                  (ltms:ltms-predicate-model))

(define-predicate has-cat-code (current-toe *cat-code*) ;;; identifies category code for TOE
                  (ltms:ltms-predicate-model))        ;;; 1- cbt, 2 - cbt spt, 3 - cbt srv spt

(define-predicate has-mission (current-toe-para *mission*)
                  (ltms:ltms-predicate-model))  ;;; predicate I believe is required to specify
                                                ;;; specified missions of TOE paragraphs
                                                ;;; should be down to paragraph level

;;; LIN SPECIFIC PREDICATES

(define-predicate is-lin (current-lin *lin*)
                  (ltms:ltms-predicate-model))  ;;; predicate to id the specifed LIN of a TOE

(define-predicate has-old-erc (current-lin *old-erc*)
                  (ltms:ltms-predicate-model))  ;;; predicate to id the old ERC of specifed LIN

(define-predicate has-new-erc (current-lin *new-erc*)
                  (ltms:ltms-predicate-model))  ;;; predicate to id the new ERC of specifed LIN

(define-predicate is-function (current-lin-func *func*)
                  (ltms:ltms-predicate-model))  ;;; predicate I believe is required to specify
                                                ;;; specified functions for pieces of equipment
                                                ;;; this is a the D & E record level
```

APPENDIX M

```
;;; -*- Mode: Joshua; Package: JOSHUA-USER; Syntax: Joshua -*-
;;; Created 4/05/90 10:39:58 by chamberlin running on SYM4 at NPS-CS.
;*****************************************************************************************
;
;   FILENAME.......... :   erc-read-toe.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   5 Apr 90
;   FILE DESCRIPTION... :  Contains the methods invoked by the "Read Current TOE"
;                          on the Interface command menu.  A single TOE record is read
;                          tested for record type, and then depending on the record
;                          instantiates global variables or is skipped.
;
;   MODIFICATIONS..... :   10 May 90 - Adjusted format of messages
;
;*****************************************************************************************

(defun read-current-toe ()
  (setf *toe-record* (setf *end-of-file* (read-line *in-file* nil)))
  (test-for-record))
(defun test-for-record ()
  (cond ((test-for-A-rec) (format t "~%~@3TA-RECORD READ !!~%")
                          (setf *title-length* (- (string-length *toe-record*) 124))
                          (get-toe-data *toe-record*))
        ((test-for-B-rec) (format t "~%~@3TB-RECORD READ !!~%")
                          (get-toe-para *toe-record*))
        ((test-for-C-rec) (format t "~%~@3TC-RECORD READ !!~%")
                          (read-current-toe))
        ((test-for-D-rec) (format t "~%~@3TD-RECORD READ !!~%")
                          (get-equip-data *toe-record*))
        ((test-for-E-rec) (format t "~%~@3TE-RECORD READ !!~%")
                          (get-equip-data *toe-record*))
        ((test-for-F-rec) (format t "~%~@3TF-RECORD READ !!~%")
                          (read-current-toe))
        ((test-for-P-rec) (format t "~%~@3TP-RECORD READ !!~%")
                          (read-current-toe))
        (t (com-flush-joshua-display)
           (format t "~8%~@59T~vEVIEWED TOE ~A~⊃
                   '(:dutch :bold :very-large) *selected-toe*)
           (format t "~2%~@57T~vO MORE RECORDS FOUND !!~⊃
                   '(:dutch :bold :very-large))
           (format t "~2%~@70T~vOE COMPLETE !!~⊃
                   '(:dutch :bold :very-large)))))

(defun test-for-A-rec ()
  (if (eq (string-search-char '#\A *toe-record* :start 41) 41) t))
(defun test-for-B-rec ()
  (if (eq (string-search-char '#\B *toe-record* :start 41) 41) t))
(defun test-for-C-rec ()
  (if (eq (string-search-char '#\C *toe-record* :start 41) 41) t))
(defun test-for-D-rec ()
  (if (eq (string-search-char '#\D *toe-record* :start 41) 41) t))
(defun test-for-E-rec ()
  (if (eq (string-search-char '#\E *toe-record* :start 41) 41) t))
(defun test-for-F-rec ()
  (if (eq (string-search-char '#\F *toe-record* :start 41) 41) t))
(defun test-for-P-rec ()
  (if (eq (string-search-char '#\P *toe-record* :start 41) 41) t))
```

APPENDIX N

```
;;; -*- Mode: Joshua; Package: USER (really JOSHUA-USER); Syntax: Joshua -*-
;;; Created 12/14/89 10:16:52 by chamberlin running on SYM4 at NPS-CS.

;*****************************************************************************************
;
;   FILENAME.......... :   erc-rules-acrft-hel.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   20 Apr 90
;   FILE DESCRIPTION... :   Rule for night operation requirement.  Found in CAA Study
;                           Report page I-23.
;
;   MODIFICATIONS..... :   16 May 90 - added 'show-rule-firing' function
;
;*****************************************************************************************

;;; SECTION 4.1 ** AIRCRAFT & HELICOPTER RULES **
;;; ---------------------------------------------
;;;

;;;
;;; CORE EQUIPMENT ** 4.1.1 **
;;;

(defrule ercp411 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-56")
  if [and [or [is-lin =CURRENT-LIN N04730]
              [is-lin =CURRENT-LIN N04596]
              [is-lin =CURRENT-LIN N15518]
              [is-lin =CURRENT-LIN N04732]
              [is-lin =CURRENT-LIN N04456]
              [is-lin =CURRENT-LIN N04982]
              [is-lin =CURRENT-LIN N23721]
              [is-lin =CURRENT-LIN W80715]
              [is-lin =CURRENT-LIN Y03104]
              [is-lin =CURRENT-LIN N05050]
              [is-lin =CURRENT-LIN A34938]
              [is-lin =CURRENT-LIN N05482]
              [is-lin =CURRENT-LIN A70349]]
          [or [is-function current-lin-func "Tactical operations"]
              [is-function current-lin-func "Maintenance Operations"]
              [is-function current-lin-func "Medical Evacuation"]]]
  then [and (setf *new-erc* 'P)
            [has-new-erc =CURRENT-LIN *new-erc*]
            (show-rule-firing)])
```

```
;;;
;;; CORE EQUIPMENT ** 4.1.2 **
;;;

(defrule erca412 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-56")
  if [and [or [is-lin =CURRENT-LIN N04730]
              [is-lin =CURRENT-LIN N04596]
              [is-lin =CURRENT-LIN N15518]
              [is-lin =CURRENT-LIN N04732]
              [is-lin =CURRENT-LIN N04456]
              [is-lin =CURRENT-LIN N04982]
              [is-lin =CURRENT-LIN N23721]
              [is-lin =CURRENT-LIN W80715]
              [is-lin =CURRENT-LIN Y03104]
              [is-lin =CURRENT-LIN N05050]
              [is-lin =CURRENT-LIN A34938]
              [is-lin =CURRENT-LIN N05482]
              [is-lin =CURRENT-LIN A70349]]
       [is-function current-lin-func "Admin-log operations"]]
  then [and (setf *new-erc* 'A)
            [has-new-erc =CURRENT-LIN *new-erc*]
            (show-rule-firing)])
```

APPENDIX O

```
;;; -*- Mode: Joshua; Package: USER (really JOSHUA-USER); Syntax: Joshua -*-
;;; Created 12/14/89 10:16:52 by chamberlin running on SYM4 at NPS-CS.

;*****************************************************************************************
;
;   FILENAME.......... :   erc-rules-binocular.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   12 Dec 89
;   FILE DESCRIPTION... :   Rule for binoculars.  Found in CAA Study
;                           Report page I-65.
;
;   MODIFICATIONS..... :   16 May 90 - added 'show-rule-firing' function
;
;*****************************************************************************************

;;; SECTION 4.8 ** BINOCULAR RULE **
;;; ------------------------------------------------
;;;

;;;
;;; BINOCULAR RULE ** 4.8.1 **
;;;

(defrule ercb481 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-65")
  if [or [is-lin =CURRENT-LIN B67081]
         [is-lin =CURRENT-LIN B67218]
         [is-lin =CURRENT-LIN B67355]
         [is-lin =CURRENT-LIN B67423]
         [is-lin =CURRENT-LIN B67492]
         [is-lin =CURRENT-LIN B67766]
         [is-lin =CURRENT-LIN B67771]]
  then [and (setf *new-erc* 'B)
            [has-new-erc =CURRENT-LIN *new-erc*]
            (show-rule-firing)])
```

APPENDIX P

```
;;; -*- Mode: Joshua; Package: USER (really JOSHUA-USER); Syntax: Joshua -*-
;;; Created 12/14/89 10:16:52 by chamberlin running on SYM4 at NPS-CS.

;*****************************************************************************************
;
;   FILENAME.......... :   erc-rules-camo-system.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   14 Dec 89
;   FILE DESCRIPTION... :   Rule for camoflague system and poles.  Found in CAA Study
;                           Report page I-64.
;
;   MODIFICATIONS..... :   16 May 90 - added 'show-rule-firing' function
;
;*****************************************************************************************

;;; SECTION 4.7 ** CAMOFLAGUE SYSTEM RULE **
;;; ---------------------------------------------------
;;;

;;;
;;; CAMOFLAGUE SYSTEM RULE ** 4.7.1 **
;;;

(defrule ercc471 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-64")
  if (or (is-lin =current-lin C89145)
         (is-lin =current-lin C89179))
  then (and (setf *new-erc* 'C)
            (tell (make-predication '(has-new-erc =CURRENT-LIN ,*new-erc*)))
            (show-rule-firing)))
```

APPENDIX Q

```
;;; -*- Mode: Joshua; Package: USER (really JOSHUA-USER); Syntax: Joshua -*-
;;; Created 4/11/90 11:32:58 by chamberlin running on SYM4 at NPS-CS.

;*****************************************************************************************
;
;   FILENAME.......... :   erc-rules-nbc-defense.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   20 Apr 90
;   FILE DESCRIPTION... :   Rule for nbc defense equipment.  Found in CAA Study
;                           Report page I-91.
;
;   MODIFICATIONS..... :   16 May 90 - added 'show-rule-firing' function
;
;*****************************************************************************************

;;; SECTION 6.1 ** NBC DEFENSE EQUIPMENT ERC RULES **
;;; ------------------------------------------------
;;;

;;;
;;; EQUIPMENT TASK --> INDIVIDUAL PROTECTION ** 6.1.1 **
;;;

(defrule erca611 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-91")
  if [or [is-lin =CURRENT-LIN M10936]
         [is-lin =CURRENT-LIN M11621]
         [is-lin =CURRENT-LIN M11895]]
  then [an  (setf *new-erc* 'A)
            (tell (make-predication '(has-new-erc =CURRENT-LIN ,*new-erc*)))
            (show-rule-firing)])
```

APPENDIX R

```
;;; -*- Mode: Joshua; Package: JOSHUA-USER; Syntax: Joshua -*-
;;; Created 4/20/90 13:49:59 by chamberlin running on SYM4 at NPS-CS.


;*****************************************************************************************
;
;   FILENAME.......... :   erc-rules-night-ops.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   20 Apr 90
;   FILE DESCRIPTION... :   Rule for night operation requirement.  Found in CAA Study
;                           Report page I-23.
;
;   MODIFICATIONS..... :   16 May 90 - added 'show-rule-firing' function
;
;*****************************************************************************************

;;; SECTION 1.3 ** UTILITY RULE FOR NIGHT OPERATIONS  **
;;; ---------------------------------------------------
;;;

;;;
;;; NIGHT OPERATIONS ** 1.3.1 **
;;;

(defrule night-ops131 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-23")
  if (or [has-branch =CURRENT-TOE 01]
         [has-branch =CURRENT-TOE 07]
         [has-branch =CURRENT-TOE 17]
         [has-branch =CURRENT-TOE 19]
         [has-branch =CURRENT-TOE 31])
  then (tell (make-predication '(has-night-ops ,=CURRENT-TOE))))
```

APPENDIX S

```
;;; -*- Mode: Joshua; Package: USER (really JOSHUA-USER); Syntax: Joshua -*-
;;; Created 12/14/89 10:16:52 by chamberlin running on SYM4 at NPS-CS.

;*****************************************************************************************
;
;   FILENAME.......... :   erc-rules-night-vision-devices.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   12 Dec 89
;   FILE DESCRIPTION... :   Rule for night vision devices.  Found in CAA Study
;                           Report page I-57.
;
;   MODIFICATIONS..... :   16 May 90 - added 'show-rule-firing' function
;
;*****************************************************************************************

;;; SECTION 4.2 ** NIGHT VISION DEVICE RULES **
;;; ------------------------------------------------
;;;

;;;
;;; INDIVIDUAL SITUATION ASSESSMENT ** 4.2.1 **
;;;

(defrule erca421 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-57")
  if [and [not [has-branch ■CURRENT-TOE 32]]
          [not [has-branch ■CURRENT-TOE 34]]
          [has-night-ops ■CURRENT-TOE]
          [or [is-lin ■CURRENT-LIN N04730]
              [is-lin ■CURRENT-LIN N04596]
              [is-lin ■CURRENT-LIN N15518]
              [is-lin ■CURRENT-LIN N04732]
              [is-lin ■CURRENT-LIN N04456]
              [is-lin ■CURRENT-LIN N04982]
              [is-lin ■CURRENT-LIN N23721]
              [is-lin ■CURRENT-LIN W80715]
              [is-lin ■CURRENT-LIN Y03104]
              [is-lin ■CURRENT-LIN N05050]
              [is-lin ■CURRENT-LIN A34938]
              [is-lin ■CURRENT-LIN N05482]
              [is-lin ■CURRENT-LIN A70349]]]
  then [and (setf *new-erc* 'A)
            [has-new-erc ■CURRENT-LIN *new-erc*]
            (show-rule-firing)])
```

```
;;;
;;; INDIVIDUAL SITUATION ASSESSMENT ** 4.2.2 **
;;;

(defrule ercb422 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-57")
  if [and [not [has-branch =CURRENT-TOE 32]]
          [not [has-branch =CURRENT-TOE 34]]
          [not [has-night-ops =CURRENT-TOE]]
          [or [is-lin =CURRENT-LIN N04730]
              [is-lin =CURRENT-LIN N04596]
              [is-lin =CURRENT-LIN N15518]
              [is-lin =CURRENT-LIN N04732]
              [is-lin =CURRENT-LIN N04456]
              [is-lin =CURRENT-LIN N04982]
              [is-lin =CURRENT-LIN N23721]
              [is-lin =CURRENT-LIN W80715]
              [is-lin =CURRENT-LIN Y03104]
              [is-lin =CURRENT-LIN N05050]
              [is-lin =CURRENT-LIN A34938]
              [is-lin =CURRENT-LIN N05482]
              [is-lin =CURRENT-LIN A70349]]]
  then [and (setf *new-erc* 'B)
            [has-new-erc =CURRENT-LIN *new-erc*]
            (show-rule-firing)])


;;;
;;; INDIVIDUAL SITUATION ASSESSMENT ** 4.2.3 **
;;;

;(defrule ercb423 (:forward
;                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-57")
;  if [and [not [has-branch =CURRENT-TOE 32]]
;          [not [has-branch =CURRENT-TOE 34]]
;          [is-lin =CURRENT-LIN "STANO"]]
;  then [and (setf *new-erc* 'B)
;            [has-new-erc =CURRENT-LIN *new-erc*]
;              (format t "~%~25@TTOE: ~A  PARA: ~A  LIN: ~A  OLD-ERC: ~A  NEW-ERC: ~A~%" *toe-nu;m* *par
 *lin* *old-erc* *new-erc*)])


;;;
;;; INDIVIDUAL SITUATION ASSESSMENT ** 4.2.4 **
;;;

;(defrule erca424 (:forward
;                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-57")
;  if [and [or [has-mission current-toe-para "24hr surveillance collection"]
;              [has-mission current-toe-para "Aerial Intelligence/Visual Observation"]]
;          [or [has-branch =CURRENT-TOE 32]
;              [has-branch =CURRENT-TOE 34]]
;          [or [is-lin =CURRENT-LIN N04732]
;              [is-lin =CURRENT-LIN N04456]
;              [is-lin =CURRENT-LIN Y03104]
;              [is-lin =CURRENT-LIN N05050]
;              [is-lin =CURRENT-LIN N05482]
;              [is-lin =CURRENT-LIN A70349]]]
;  then [and (setf *new-erc* 'A)
;            [has-new-erc =CURRENT-LIN *new-erc*]
;              (format t "~%~25@TTOE: ~A  PARA: ~A  LIN: ~A  OLD-ERC: ~A  NEW-ERC: ~A~%" *toe-nu;m* *par
 *lin* *old-erc* *new-erc*)])
```

```
;;;
;;; INDIVIDUAL SITUATION ASSESSMENT ** 4.2.5 **
;;;

;(defrule ercb425 (:forward
;                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-57")
;  if [and [or [not [has-mission current-toe-para "24hr surveillance collection"]]
;              [not [has-mission current-toe-para "Aerial Intelligence/Visual Observation"]]]
;          [or [has-branch =CURRENT-TOE 32]
;              [has-branch =CURRENT-TOE 34]]
;          [or [is-lin =CURRENT-LIN N04732]
;              [is-lin =CURRENT-LIN N04456]
;              [is-lin =CURRENT-LIN Y03104]
;              [is-lin =CURRENT-LIN N05050]
;              [is-lin =CURRENT-LIN N05482]
;              [is-lin =CURRENT-LIN A70349]]]
;  then [and (setf *new-erc* 'A)
;            [has-new-erc =CURRENT-LIN *new-erc*]
;             (format t "~%~25@TTOE: ~A  PARA: ~A  LIN: ~A  OLD-ERC: ~A  NEW-ERC: ~A~%" *toe-num;* *para
*lin* *old-erc* *new-erc*)]))
```

APPENDIX T

```
;;; -*- Mode: Joshua; Package: USER (really JOSHUA-USER); Syntax: Joshua -*-
;;; Created 4/11/90 11:32:58 by chamberlin running on SYM4 at NPS-CS.

;*****************************************************************************************
;
;   FILENAME.......... :   erc-rules-weapons.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   11 Apr 90
;   FILE DESCRIPTION... :   Rule for all weapon systems.  Found in CAA Study
;                           Report page I-81.
;
;   MODIFICATIONS..... :   16 May 90 - added 'show-rule-firing' function
;
;*****************************************************************************************

;;; SECTION 5.3 ** WEAPON & ASSOCIATED EQUIPMENT ERC RULES **
;;; -------------------------------------------------------
;;;

;;;
;;; EQUIPMENT TASK --> CORE EQUIPMENT ** 5.3.1 **
;;;

(defrule ercp531 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-81")
  if [or [is-lin =CURRENT-LIN T13168]
         [is-lin =CURRENT-LIN T13169]
         [is-lin =CURRENT-LIN T13174]
         [is-lin =CURRENT-LIN Z77258]
         [is-lin =CURRENT-LIN F40307]
         [is-lin =CURRENT-LIN J81750]
         [is-lin =CURRENT-LIN C76335]
         [is-lin =CURRENT-LIN F60462]
         [is-lin =CURRENT-LIN K56981]
         [is-lin =CURRENT-LIN K57392]
         [is-lin =CURRENT-LIN K57667]
         [is-lin =CURRENT-LIN K57803]
         [is-lin =CURRENT-LIN K57821]
         [is-lin =CURRENT-LIN H57505]
         [is-lin =CURRENT-LIN Z33628]]
  then [and (setf *new-erc* 'P)
            (tell (make-predication '(has-new-erc =CURRENT-LIN ,*new-erc*)))
            (show-rule-firing)]]
```

```
;;; (setf db-obj-1 (tell (make-predication '(has-new-erc =CURRENT-LIN ,*new-erc*))))
;;;
;;; EQUIPMENT TASK --> CORE EQUIPMENT ** 5.3.2 **
;;;

(defrule erca532 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-81")
  if [and [or [has-branch =CURRENT-TOE 07]
              [has-branch =CURRENT-TOE 31]]
          [or [is-lin =CURRENT-LIN R95035]
              [is-lin =CURRENT-LIN R94977]
              [is-lin =CURRENT-LIN N96741]
              [is-lin =CURRENT-LIN R91244]
              [is-lin =CURRENT-LIN M09009]
              [is-lin =CURRENT-LIN P98152]
              [is-lin =CURRENT-LIN Z13153]
              [is-lin =CURRENT-LIN B49004]
              [is-lin =CURRENT-LIN B49272]
              [is-lin =CURRENT-LIN B68790]
              [is-lin =CURRENT-LIN M67939]
              [is-lin =CURRENT-LIN M02114]
              [is-lin =CURRENT-LIN M68282]
              [is-lin =CURRENT-LIN M92420]
              [is-lin =CURRENT-LIN G96797]
              [is-lin =CURRENT-LIN Z13322]
              [is-lin =CURRENT-LIN J97983]
              [is-lin =CURRENT-LIN L91975]
              [is-lin =CURRENT-LIN L92112]
              [is-lin =CURRENT-LIN L92260]
              [is-lin =CURRENT-LIN L92386]
              [is-lin =CURRENT-LIN R96484]]]
  then [and (setf *new-erc* 'A)
            (tell (make-predication '(has-new-erc =CURRENT-LIN ,*new-erc*)))
            (show-rule-firing)])

;;;
;;; EQUIPMENT TASK --> UNIT DEFENSE ** 5.3.3 **
;;;

(defrule ercp533 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-81")
  if [and [not [has-branch =CURRENT-TOE 07]]
          [not [has-branch =CURRENT-TOE 31]]
          [has-cat-code =CURRENT-TOE 1]
          [or [is-lin =CURRENT-LIN R95035]
              [is-lin =CURRENT-LIN R94977]
              [is-lin =CURRENT-LIN N96741]
              [is-lin =CURRENT-LIN R91244]
              [is-lin =CURRENT-LIN M09009]
              [is-lin =CURRENT-LIN P98152]
              [is-lin =CURRENT-LIN Z13153]
              [is-lin =CURRENT-LIN M67939]
              [is-lin =CURRENT-LIN M02114]
              [is-lin =CURRENT-LIN M68282]
              [is-lin =CURRENT-LIN M92420]
              [is-lin =CURRENT-LIN G96797]
              [is-lin =CURRENT-LIN Z13322]
              [is-lin =CURRENT-LIN J97983]
              [is-lin =CURRENT-LIN L91975]
              [is-lin =CURRENT-LIN L92112]
              [is-lin =CURRENT-LIN L92260]
              [is-lin =CURRENT-LIN L92386]
              [is-lin =CURRENT-LIN R96484]]]
  then [and (setf *new-erc* 'A)
            (tell (make-predication '(has-new-erc =CURRENT-LIN ,*new-erc*)))
            (show-rule-firing)])
```

```
;;;
;;; EQUIPMENT TASK --> UNIT DEFENSE ** 5.3.4 **
;;;

(defrule ercp534 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-81")
  if [and [not [has-cat-code ■CURRENT-TOE 1]]
          [or [is-lin ■CURRENT-LIN R95035]
              [is-lin ■CURRENT-LIN R94977]
              [is-lin ■CURRENT-LIN N96741]
              [is-lin ■CURRENT-LIN R91244]
              [is-lin ■CURRENT-LIN M09009]
              [is-lin ■CURRENT-LIN P98152]
              [is-lin ■CURRENT-LIN Z13153]
              [is-lin ■CURRENT-LIN M92420]
              [is-lin ■CURRENT-LIN G96797]
              [is-lin ■CURRENT-LIN J97983]
              [is-lin ■CURRENT-LIN L91975]
              [is-lin ■CURRENT-LIN L92112]
              [is-lin ■CURRENT-LIN L92260]
              [is-lin ■CURRENT-LIN L92386]
              [is-lin ■CURRENT-LIN R96484]]]
  then [and (setf *new-erc* 'B)
            (tell (make-predication '(has-new-erc ■CURRENT-LIN ,*new-erc*)))
            (show-rule-firing)])

;;;
;;; EQUIPMENT TASK --> CORE EQUIPMENT  ** 5.3.5 **
;;;

(defrule erca535 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-81")
   if [and [has-branch ■CURRENT-TOE 19]
           [or [is-lin ■CURRENT-LIN R95035]
               [is-lin ■CURRENT-LIN R94977]
               [is-lin ■CURRENT-LIN N96741]
               [is-lin ■CURRENT-LIN R91244]
               [is-lin ■CURRENT-LIN M09009]
               [is-lin ■CURRENT-LIN P98152]
               [is-lin ■CURRENT-LIN Z13153]
               [is-lin ■CURRENT-LIN M92420]
               [is-lin ■CURRENT-LIN G96797]
               [is-lin ■CURRENT-LIN J97983]
               [is-lin ■CURRENT-LIN L91975]
               [is-lin ■CURRENT-LIN L92112]
               [is-lin ■CURRENT-LIN L92260]
               [is-lin ■CURRENT-LIN L92386]
               [is-lin ■CURRENT-LIN R96484]]]
  then [and (setf *new-erc* 'A)
            (tell (make-predication '(has-new-erc ■CURRENT-LIN ,*new-erc*)))
            (show-rule-firing)])

;;;
;;; EQUIPMENT TASK --> UNIT DEFENSE  ** 5.3.6 **
;;;

(defrule erca536 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-81")
  if [and [not [has-branch ■CURRENT-TOE 07]]
          [not [has-branch ■CURRENT-TOE 31]]
          [or [is-lin ■CURRENT-LIN B49004]
              [is-lin ■CURRENT-LIN B49272]]]
  then [and (setf *new-erc* 'B)
            (tell (make-predication '(has-new-erc ■CURRENT-LIN ,*new-erc*)))
            (show-rule-firing)])
```

```
;;;
;;; EQUIPMENT TASK --> WEAPON FIRE-LAYING DEVICES ** 5.3.9 **
;;;

;;;(defrule erca539 (:forward
;;;                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-82")
;;;  if [and [is-lin =CURRENT-LIN wpn-firing-laying-device]
;;;          [is-function =CURRENT-LIN-func 11]]
;;;  then [and (setf *new-erc* 'A)
;;;            (tell (make-predication '(has-new-erc =CURRENT-LIN ,*new-erc*)))
;;;            (show-rule-firing)])

;;;
;;; EQUIPMENT TASK --> WEAPON FIRE-LAYING DEVICES ** 5.3.10 **
;;;

;;;(defrule ercb5310 (:forward
;;;                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-82")
;;;  if [and [is-lin =CURRENT-LIN wpn-firing-laying-device]
;;;          [is-function =CURRENT-LIN-func 12]
;;;  then [and (setf *new-erc* 'B)
;;;            (tell (make-predication '(has-new-erc =CURRENT-LIN ,*new-erc*)))]])

;;;
;;; EQUIPMENT TASK --> ENHANCEMENT/SUPPORT EQUIPMENT  ** 5.3.8 **
;;;

(defrule ercb538 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-82")
  if [or [is-lin =CURRENT-LIN M92362]
         [is-lin =CURRENT-LIN M74364]
         [is-lin =CURRENT-LIN M75577]
         [is-lin =CURRENT-LIN M75714]]
;;;      [is-lin =CURRENT-LIN other-assoc-item]]]
  then [and (setf *new-erc* 'B)
            (tell (make-predication '(has-new-erc =CURRENT-LIN ,*new-erc*)))
            (show-rule-firing)])
```

APPENDIX U

```
;;; -*- Mode: Joshua; Package: USER (really JOSHUA-USER); Syntax: Joshua -*-
;;; Created 12/14/89 10:16:52 by chamberlin running on SYM4 at NPS-CS.

;*****************************************************************************************
;
;   FILENAME.......... :   erc-rules-wristwatch.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   14 Dec 89
;   FILE DESCRIPTION... :   Rule for wristwatch.  Found in CAA Study
;                           Report page I-66.
;
;   MODIFICATIONS..... :   16 May 90 - added 'show-rule-firing' function
;
;*****************************************************************************************

;;; SECTION 4.9 ** WRIST WATCH RULE **
;;; -------------------------------------------------
;;;

;;;
;;; WRIST WATCH RULE ** 4.9.1 **
;;;

(defrule ercc491 (:forward
                  :documentation "Basis: Study Report CAA-SR-88-14 dtd June 88, p. I-66")
  if [is-lin ■CURRENT-LIN wristwatch]
  then [and (setf *new-erc* 'C)
            [has-new-erc ■CURRENT-LIN *new-erc*]
            (show-rule-firing)])
```

APPENDIX V

```
;;; -*- Mode: Joshua; Package: JOSHUA-USER; Syntax: Joshua -*-
;;; Created 4/25/90 10:49:28 by chamberlin running on SYM4 at NPS-CS.

;*****************************************************************************************
;
;   FILENAME.......... :   erc-toe-list.lisp
;   AUTHOR............ :   Cpt Thomas E. Chamberlin
;
;   DATE CREATED...... :   25 Apr 90
;   FILE DESCRIPTION... :   All TOE's downloaded into Symbolics machine to run thru
;                           expert system must have their filenames placed here under
;                           'toe-list' to be made available for system.  The interface
;                           command 'Select TOE' invokes this function.
;
;   MODIFICATIONS..... :
;
;*****************************************************************************************

(setf toe-list '("06037L200"
                 "17442L000"
                 "17477L000"
                 "17486L000"
                 "17487L000"
                 "17487test"
                 "34286L000"
                 "34287L000"
                 "34288L000"
                 "34289L000"))

(defun select-toe ()
  (setf *selected-toe*
        (dw:menu-choose-from-set toe-list 'string
                                 :prompt "SELECT A TOE"
                                 :center-p t
                                 :momentary-p nil
                                 :temporary-p t
                                 :near-mode '(:point 600 400)
                                 :minimum-width 11
                                 :minimum-height 3
                                 :character-style
                                 '(:dutch :bold :normal))))
```

LIST OF REFERENCES

1. Documentation Group of Symbolics, Inc., *User's Guide to Basic Joshua*, CSA Press, 1988.

2. Documentation Group of Symbolics, Inc., *Programming the User Interface*, v. 7A, CSA Press, 1986.

3. U.S. Army Concepts Analysis Agency, Study Report CAA-SR-88-14, *Equipment Readiness Code Rule System(ERCRULES)*, Government Printing Office, Washington DC, 1988.

4. Department of Army Headquarters, Deputy Chief of Staff for Operations, *Army Regulation 220-1, Field Organizations, Unit Status Reporting*, Government Printing Office, Washington, DC, 1988.

5. Bowerman, R. G., and Glover, D. E., *Putting Expert Systems Into Practice*, Van Nostrand Reinhold Co., 1988.

6. Nielsen, N. R., and Walters, J. R., *Crafting Knowledge-Based Systems*, John Wiley and Sons, 1988.

7. Rowe, N. C., *Artificial Intelligence Through Prolog*, Prentice-Hall, Inc., 1988.

8. Documentation Group of Symbolics, Inc., *Joshua Reference Manual*, CSA Press, 1988.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Curriculum Officer, Code 37 Computer Technology Naval Postgraduate School Monterey, CA 93940-5000	1
5.	HQ TRADOC ATCD-O (Mr. Richards) Fort Monroe, VA 23651-5000	12
6.	Professor Se-Hung Kwak Department of Computer Science, Code 52KW Naval Postgraduate School Monterey, CA 93940-5000	5
7.	Major George Thurmond II Department of Computer Science, Code 52TH Naval Postgraduate School Monterey, CA 93940-5000	3
8.	Captain Thomas E. Chamberlin Route #1, Box 393 Catlett, VA 22019	4